Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 129 (2014) 546 – 553

Procedia Social and Behavioral Sciences

ICIMTR 2013

International Conference on Innovation, Management and Technology Research, Malaysia, 22 – 23 September, 2013

Secure Feature Driven Development (SFDD) Model for Secure Software Development

Adila Firdaus[a], Imran Ghani[b*], Seung Ryul Jeong[c]

[a,b]*Department of Software Engineering, Faculty of Computer Science and Information Systems Universiti Teknologi Malaysi*
[c]*School of Management Information Systems, Graduate School of Business IT, Kookmin University, Seoul, South Korea.*

Abstract

This paper introduces an enhanced Feature Driven Development (FDD) model for secure software development. In fact, the enhanced model is based on our previous study and its findings which concluded that existing FDD poses limitations to develop secure software. Thus, an enhanced FDD that supports secure software development is proposed. We have implemented this new FDD model and conducted a case study to compare the level of security in the undergraduate and postgraduate level students. The paper illustrates that agility of FDD is not affected significantly, even after adding new phases.

Keywords: Agile Methodology; Security; Software Engineering ; Feature Driven Development

1. Introduction

Traditional agile software development practices considered that the true rival to software development process is complexity and size of the software (Highsmith, 2002; Hunt, 2006; Royce, 2009; Lau, 1998; Rakkhis, 2012; Rohen, 2002). However, a number of recent studies (Azham, Imran & Norafida, 2011; Imran & Nur Izati, 2013; Adila, Imran & Nor Izaty, 2013; Abdullah, Adila, Seung & Imran, 2013)

* Corresponding author. Tel. +607-553234
E-mail address: imransaieen@gmail.com

doi:10.1016/j.sbspro.2014.03.712

illustrated a more critical factor ignored by agile methods, i.e., software Developing software efficiently but not securely has enormous impacts such as loss of data, loss of reputation, loss of customers' confidence and so forth. Thus, develop secure software in efficient way, is an emerging issue. The above studies clearly describe that the existing agile methods, such Scrum, XP, DSDM, and FDD pose limitations, when it comes to develop secure software. In order to address this issue the existing agile methods needs to be revisited and enhances so that they could provide new phases, sub-phases, practices and roles related to secure software development. Recently, we enhanced Scrum model that could support secure software development. In the continuity of our research, this paper focuses on one of the agile development process, called Feature Driven Development (FDD).

Before going into the detailed limitations of existing FDD, it is appropriate to discuss the security principles that should be followed while developing secure software.

2. Principles Of Secure Software Development

Though, there are a number of software security models (Musa, et al., 2011; Spruit & Looijen, 1996; Riley et al., 2010; Ren, et al., 2005; Jones, 2007; Sharma & Trivedi, 2007; El-Attar, 2012), however we focus on the general security principles. The reason of this choice is that the security is implemented throughout the lifecycle of FDD agile method. In order to gain a full understanding and handling of the security principles, we considered the following security principals, which have been ignored in the traditional agile software development methods, including FDD.

a) The Principle of Least Privilege – Only the minimum necessary rights should be assigned to a subject that requests access to a resource and should be in effect for the shortest duration necessary. Other security activities that can be implement here is Minimize the damage, minimize interaction between privileged programs, password management, limit the access to database and restrict the access time.
b) The Principle of Failing Securely – When a system fails, it should do so securely. This behavior typically includes several elements: secure defaults the default is to deny access Security activities that can be implement here are Grant Access when not Explicitly forbidden, Ease of use, In case of mistake, access denied, No default passwords, No sample users, Files are write protected, owned by root, Error message generic, Error message information in log files.
c) The Principle of Securing the Weakest Link – Attackers are more likely to attack a weak spot in a software system than to penetrate a heavily fortified component. For example, some cryptographic algorithms can take many years to break, so attackers are unlikely to attack encrypted information communicated in a network.
d) The Principle of Defense in Depth – Layering security defenses in an application can reduce the chance of a successful attack. Incorporating redundant security mechanisms requires an attacker to circumvent each mechanism to gain access to a digital asset. For example, we need to use multiple layered protection software.
e) The Principle of Separation of Privilege – A system should ensure that multiple conditions are met before it grants permissions to an object. Checking access on only one condition may not be adequate for enforcing strong security. Compartmentalizing software into separate components that require multiple checks for access can inhibit an attack or potentially prevent an attacker from taking over an entire system.
f) The Principle of Economy of Mechanism – One factor in evaluating a system's security is its complexity. Keep design simple and remove unnecessary data and code.

g) The Principle of Least Common Mechanism – Avoid having multiple subjects share those mechanisms that grant access to a resource. For example security activities can be add here is reduce potentially dangerous information flow, reduce possible interaction and make it more flexible.
h) The Principle of Complete Mediation – A software system that requires access checks to an object each time a subject requests access, especially for security Critical objects, decreases the chances that the system will mistakenly give elevated permissions to that subject. For example we need to make identification of source action. Make sure user is talking to authentication program, Safe login, Window control+alt+delete, Secure interface, Input validation. Do not authenticate based on IP source, email sender can be forged, hidden fields, and safely load.

Several researches (Alnatheer, et al., 2010; Keramati & Seyed-Hassan, 2008; Ejan & Bengt, 2011; Ergogan, et al., 2010) on existing agile methodologies show that they do not provide guidelines to implement such security principles. The following section describes the security related limitations of FDD.

3. Security Related Limitations of Fdd

The main reason why this study has been conducted is due to the issues that lingering around in the mind of agile teams who adopts FDD. The team members always ask question that is it possible to keep the process agile and produce a secured system? After doing some analysis on the advantages and disadvantages of FDD, here are a few issues that were identified:

a) Security usually defined as non-functional requirements. That's why the security only been applied after the whole features of the system have been implemented. Due to that action, it will drag the completion date of the software
b) The software developer neglects question about each feature that was defined in the system is acceptably secured.
c) FDD defines feature based on the users' requirements. However not all clients are well defined with the software security features.
d) There is a lack of defining the security methods, techniques, process, or maybe phases that available in the existing FDD model.
e) There are no security roles that have been defined in FDD that can help to assist in identifying security elements inside the software or the system's features itself.

4. Proposed Conceptual Model Of Secure Fdd (Sfdd)

After a few case studies Adila, et al., (2013) that helped to identify the limitations in the existing FDD model, we attempt to propose a conceptual Secure Feature Driven Development (SFDD) model. In this enhanced FDD model, we made a number of changes.

- The first change we made is to combine two existing phases Design by Feature phase and Built a Feature List into one Build and Design Features
- The second change is about adding new elements inside the phases called In-Phase Security.
- The third change is about adding two additional phases named as Build Security by Feature and Test Security by Feature. This feature is called After-Phase Security.
- The fourth change is about introducing a new role called Security Master.

All of these new improvements in FDD methods are explained in the next- subsection.

4.1. Changes of Phases

As we know that the existing FDD model there are 5 main phases. Based on the case study, we find out that *Design by Feature* phase should not be a phase on its own as it joins the *Build a Feature List*. Both these phases need to be combined in one phase; we call it *Build and Design Features* (Fig 1). There are two reasons for this:

a) It saves precious time if after building the features list, it can directly design the feature and plan the implementation based on the design. A better planning can be done if we can see the whole design of the features in one phase.

b) This is a good practice for less experienced development team where they have less knowledge in estimating the feature just by looking the list of feature compare to looking the whole design of it.

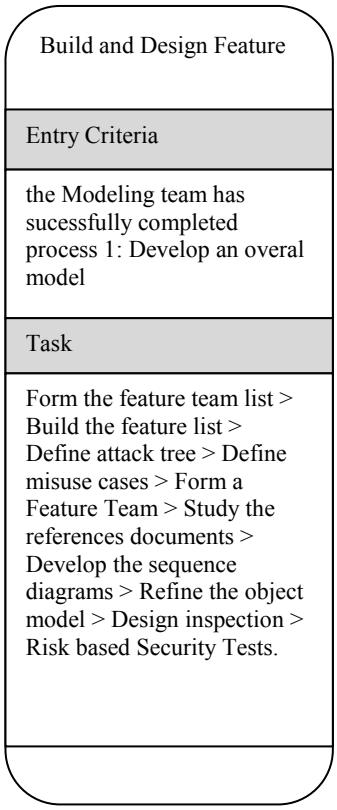

Fig 1: Combined Build and Design Feature

4.2. In-Phase Security

This aspect includes necessary security process within the FDD phases. By adding security aspect inside the phase, red coloured features in Fig 1 and Fig 2, the FDD team can identify maximum risk of a system

based on the user requirements or the overall model. Based on our proposed model (Fig 3), all phases include security aspect. It also means that security aspect has been introduced in all the phases of existing FDD model (Fig 3).

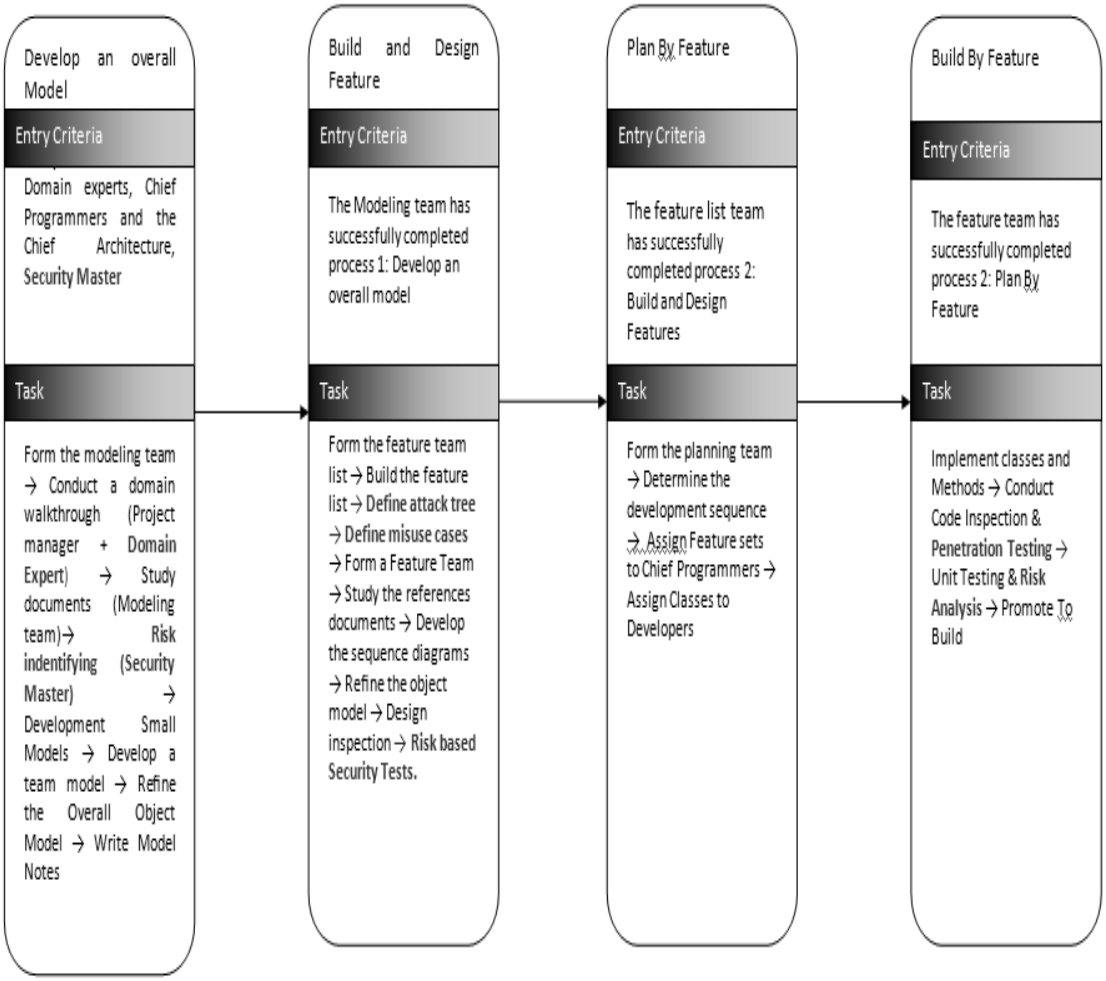

Fig 2: SFDD Model

4.3.After-Phase security

Unlike the In-Phase security, the security aspect has been defined after a phase, called After-Phase security. For this, we proposed two new phases, Build Security By Feature and Test Security by Feature.

The two new phases are shown in grey colour in Fig 3. These phases are quite suitable for experienced as well as new and less experienced team to develop and test secure software.

Based on these modifications, there are six phases altogether that start with Develop an Overall Model, Build and Design Features, Build Security By Features, Plan By Features, Build By Features and Test Security By Features.

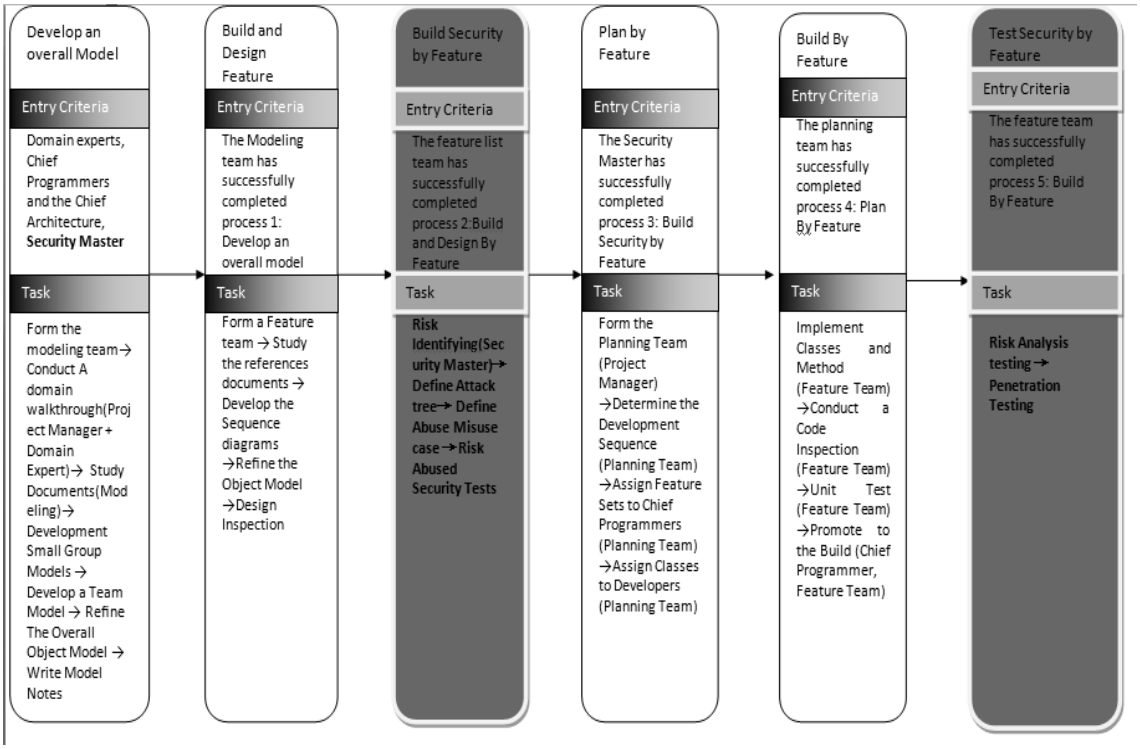

Fig 3: After-Phase Security

4.4.Security Master

Based on our case studies at postgraduate and undergraduate level projects, it has been identified that there is a need for security focused team member who could guide the team about secure design, secure development and security testing. Hence, a new role known as Security Master (SM) has been introduced. The SM is in charge of the security during the FDD lifecycle. The SM marks the selected features in first phase. 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 for the attention of the developers. Testing of those features is verified in phases by the security master.

5. Conclusions And Future Work

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 FDD method. we were able to discover the relationship between the security principles and security in each of the FDD phases. The second issue, then, was to enhance the FDD model in relation to security. The enhanced SFDD model that we have proposed will be evaluated in the requirement, development and testing phases. These research objectives will be completed successfully in our ongoing research. An agile team at university will present evaluations and feedback in regards to the enhancement model. The findings of our ongoing research will be shared in near future.

Acknowledgements

This project is funded by Ministry of Science, Technology and Innovation (MOSTI) Malaysia, under the Vote Number: 4S028.

References

Highsmith, J., (2002).What is agile software development?. In Boston: *Crosswalk*.

Hunt, J., (2006). Feature-Driven Development 9.1. In *Agile Software Construction*, 161–181.

Royce, W., (2009).*Improving Software Economics*. in white paper. IBM.

Lau, O. (1998). The Ten Commandments of Security. *Computers and Security*. 17, 119-123.

Rakkhis. (2012) *Agile!=Security*.[Online]. Available : http://www.rakkhis.com/2011/06/agile-security.html

Azham, Z., Imran Ghani, Norafida Ithnin. (2011). Security backlog in scrum security practices. *5th Malaysian Conference in Software Engineering* (MySEC).

Rhoden, E., (2002). People and processes — the key elements to information security.*Computer Fraud and Security*, 14-15.

Musa, B.S., Norwawi, N.M., Selamat, M.H., Sharif, K. Y., (2011). Improved extreme programming. *IEEE Symposium on Computers & Informatics*.

Spruit, M.E.M. and Looijen, M., (1996). IT security in Dutch practice. *Computers and Security*. vol. 15(2), 157-170.

Riley, R., Jiang, X., Xu, D., (2010). An architectural approach to preventing code injection attacks. *IEEE Transactions On Dependable And Secure Computing*, 7, 4.

Ren, J., Taylor, R., Dourish, P., Redmiles, D., (2005). Towards an architectural treatment of software security: a connector-centric approach software engineering for secure systems. *Building Trustworthy Applications*.

Jones, A., (2007). A framework for the management of information security risk. *BT Technology* .

Sharma, V.S., Trivedi, K.S., (2007).Quantifying software performance, reliability and security:An architecture-based approach. *The Journal of Systems and Software 80*. 493–509.

El-Attar, M., (2012). A framework for improving quality in misuse case models. *Business Process Management Journal*. 18, 2.

Alnatheer, A., Gravell, A.M., and Argles, D., (2010). Agile security issues : an empirical study, no. 2005. 4503.

Hossein Keramati, and Seyed-Hassan Mirian-Hosseinabadi, (2008). Integrating software development security activities with agile methodologies. *IEEE*. 978-1-14244-1968-5/08.

Beznosov, K., and Kruchten, P., (2004). Towards agile security assurance. *Proceedings of the 2004 Workshop on New Security Paradigms*.

Dejan Baca, and Bengt Carlsson, (2011). Agile development with security engineering activities. *Proceeding, ICSSP '11 Proceedings of International Conference on Software and Systems Process*.

Gencer Erdogan, Per Hakon Meland, and Derek Mathieson. (2010). Security testing in agile web application development - a case study using the east methodology, *XP 2010, LNBIP 48*, 14-27, 2010. Springer-Verlag Berlin Heidelberg.

Imran Ghani, and Nur Izaty Bt Yasin (2013), Software security engineering in extreme programming methodology: a systematic literature review, Journal Science International Lahore, 25(2), 215-221.

Adila Firdaus, Imran Ghani, and Nor Izzaty Mohd Yasin (2013), Developing websites using feature driven development: a case study, Journal of Clean Energy Technologies, 1(4).

Abdullahi Sani, Adila Firdaus, Seung Ryul Jeong and Imran Ghani. (2013). A review on software development security engineering using dynamic system method (DSDM). International Journal of Computer Applications 69(25), 33-44.