A Review on Software Development Security Engineering using Dynamic System Method (DSDM)

Abdullahi Sani, Adila Firdaus
Faculty of Computing
Universiti Teknologi Malaysia

Seung Ryul Jeong
School of Management
Information Systems (MIS),
Kookmin University, Korea.

Imran Ghani
Faculty of Computing
Universiti Teknologi Malaysia

ABSTRACT
Agile methodology such as Scrum, Extreme Programming (XP), Feature Driven Development (FDD) and the Dynamic System Development Method (DSDM) have gained enough recognition as efficient development process by delivering software fast even under the time constrains. However, like other agile methods DSDM has been criticized because of unavailability of security element in its four phases. In order to have a deeper look into the matter and discover more about the reality, we conducted a literature review. Our findings highlight that, in its current form, the DSDM does not support developing secure software. Although, there are a few researches on this topic about Scrum, XP and FDD but, based on our findings, there is no research on developing secure software using DSDM. Thus, in our future work we intend to propose enhanced DSDM that will cater the security aspects in software development.

Keywords
Agile Development; Software Security; Software Engineering; Dynamic System Development Method; DSDM

1. INTRODUCTION
The DSDM software development approach that provides a framework for building and maintaining systems, meets tight time schedule through the use of incremental and iterative prototyping in a controlled project environment [1]. On the other hand, According to the computer Emergency Response Team (CERT) statistics[3].There had been a considerable increase in security related software vulnerabilities reported over the last few years. However, like other agile methods, the existing DSDM does not provide any phase or sub-phase to address security issue in software development. In general, one of the most important reasons why the agile methods ignore security issue of software may come from the misconception that security delays development process [2].Despite this misconception, security remains one of the most important non-functional requirements of a software system. Though, recently, a few efforts have made in order to address the security in software development, such efforts using agile models like Scrum, XP. Some of such efforts have been published [6][9][30][31][32][33].

However, based on the literature review, we found that there is a small amount of research conducted on developing secure software using DSDM. In order to have a deeper look into the fact, this paper presents the concepts of DSDM, its principles, techniques, practices, general security principles, limitations of DSDM in terms of addressing security, and the analysis of literature review. Thus, it is appropriate to commence with the concept of DSDM.

2. DSDM AND PRINCIPLES
The basic concept of DSDM is that the time and resource are adjusted, so that the agility feature of DSDM is satisfied. Basically, DSDM has four main phases (Figure 1). The four main phases are feasibility, functional model iteration, design and build iteration and implementation. Then each phase has several sub-phases as mentioned below.

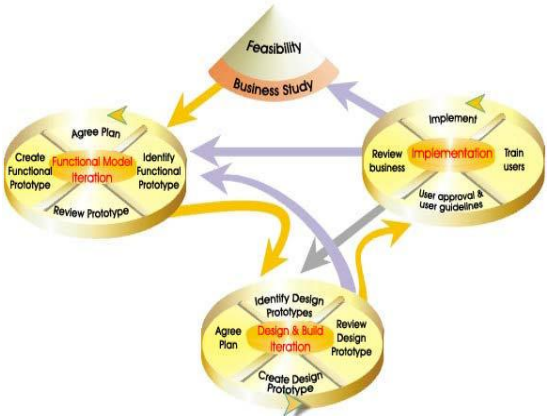

Fig 1:DSDM phases and sub-phases

Basically, the DSDM is mainly divided into two major phases, Pre-project phase and Post-project phase.

•Pre-project: This phase concerns the decision to set up the project in the first place that is business/Board decision. There are several sub-phases in this phase.

• Feasibility Study: This sub-phase is a short study of a few days or a few weeks. In this phase fairly conventional questions are asked such as "*is this project worth doing*?",and "*is this technically possible?*"
• Business Study: This sub-phase identifies business processes to be automated; high level functions and non-functional requirements; prioritises functions; outlines system architecture; produces outline plan for development, feasibility and business studies are same as Inception in the Unified Process.
• Functional Model Iteration –In this sub-phase analysis models and software components are built, which are based on the high-level models defined in the Business Study phase, equivalent to Elaboration.
• System Design and Build Iteration: This sub-phase is the system engineering phase; the main product here is the tested system, equal to Construction
• Implementation: This phase is the cutover from the development environment to the operational environment, including training the users and handing over the system to them is equal to Transition.

• Post-project: This phase maintains the post-implementation review to assess success of project. Question like "*has it delivered intended benefits*?" are answered.

The DSDM is based on a few agile principles. Most of these principles are common in other agile methods i.e., Scrum, XP, and FDD also. The DSDM principles are divided into nine where each principle works hand in hand with the other. The principles of DSDM are as follows.

1. Active user involvement is unavoidable.
2. Always DSDM teams must be empowered to make decisions.
3. The main concern is on frequent delivery of products.
4. Suitability for business purpose is the essential criteria for acceptance of deliverable.
5. In order to converge on accurate business solution, iterative and incremental development is necessary.
6. During development process all changes are reversible.
7. The baseline of requirements is at a high level.
8. Testing is integrated throughout the life-cycle.
9. The collaborative and cooperative approach between all stakeholders is essential.

Since, the focus of this paper is to study strengths and limitations of existing DSDM, thus it is appropriate to discuss that "*what actually is important while developing secure software*?" In order to answer this question, we need to look into the principles of developing software security (Section 3) recommended by [4].

2.1 DSDM Techniques

DSDM rely heavily on techniques to develop an application. However, various techniques can work together to addressed the gap among different aspects (requirements prioritization, stakeholders' concerns, project management and so on) of development. Below are some of the techniques used in DSDM development.

- MoSCoW Prioritization [13].Prioritization is essential because things that are important must be considered before things that are less, because there is no enough time to do everything. The functionality is categorized according to its importance:
  - Must Have – The consideration is given to the most important things and that are fundamental to the system.
  - Should Have–The important things for the business solution.
  - Could Have – The things that are useful but system can be developed without them for a while.
- Won't Have -The things that can easily wait until later.
- Prototyping: DSDM uses prototypes heavily to make sure that all interested parties have a clear picture of the various aspects of the system.
- Facilitated Workshops: Workshops facilitation allow for the following benefits:
- The environment must be ideal for the formation of ideas and those ideas are quick and balance growth.
- Awareness of decisions made by all interested parties by other interested parties.
- Wider range of stakeholders can make decisions.
- Quick and accurate decisions are made.
- Time-Boxing: Project management which traditionally, uses milestones to harmonize on a deliverables to a given point in time. Whereas milestones work well enough, the much powerful tool to achieve the same result is time-box[8].A time-box is an interval, usually no longer then 6 weeks, where a given set of tasks should be achieved. The reason for the relatively low duration of time-boxes is the based on the fact, that humans give much more accurate estimates in the near-future involving a small set of tasks, while estimates into the distant future involving large sets of tasks turn out to have hefty errors. Time-boxes can contain several tasks, and at the end need to deliver a product. Milestones also suffer from having a fixed deliverable, while time-boxes are subject to change, since the tasks are defined, not the necessarily the deliverable, which can change if prioritization shifts during the time-box iteration, allowing for rapid response to business needs. In short DSDM rather drops functionality in favour of delivering in time [8].

Rather than being just a process model, DSDM is a framework for software development, which includes project management, estimating, prototyping, time boxing, configuration management, prioritized requirements, implementing, testing, quality assurance, roles and responsibilities of users and IT staff, team structures and tool environments. Thus, the software security is not the concern of existing DSDM. The following section briefly describes the limitations of DSDM to develop secure software.

2.2 DSDM and Limitations in Secure Software Development

As mentioned above in the introduction section that the typical focus of DSDM and its phases is to manage change in requirements during the different phases of the software development and frequent delivery of software. Thus, the existing DSDM does not offer any guidelines to develop

secure software. The absence of security guidelines is due to the main focus on the followings

1. Agile process models are known to believe working software as the primary measure of success, and do not pay attention to security features.
2. Users' involvement in the project is essential in DSDM process model. However, in its current form, DSDM does not include any particular role of security stakeholders.
3. In DSDM, there is absence of any phase or phase in order to cater the security issues while collecting/analysing requirements, designing, or implementing of software.

In general the following are well-known security principles for developing secure software, using a traditional software development lifecycle (SDLC) or agile model.

3. Software Security Principles

The backbone of any software security is the security principles that are the domain in this context. For a full understanding and treatment of the security, we considered the following security principles [4], which have been ignored in most of the traditional and agile software development methods.

- The Principle of Failing Securely – Whenever there is a system failure, it should do so securely. This characteristic feature typically includes various elements: the secure defaults that deny access [5]. 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.
- The Principle of Defence in Depth – In this case layering security defences in an application can reduce the chance of a successful attack. Integrating security mechanisms such as redundant security 415 mechanisms needs an attacker to bypass each requires mechanism to get access to a digital asset. For instance, we need to use multiple layered protection software.
- The Principle of Separation of Privilege – Before granting permissions to an object, the system should make sure that multiple conditions are met. Checking access on only one condition may not be enough for enforcing strong security. To prevent attacker from taking over an entire system, software development process should be divided into components that require multiple checks for access.
- The Principle of Least Privilege –The right should be assigned only to the minimum subject that request access to a resource and should be within shortest possible time. Other security activities that can be implement here is Minimize the damage, minimize interaction between privileged programs, password management, restrict the access time and limit the access to database.
- The Principle of Securing the Weakest Link – It is more likely for attackers to attack weak part than to penetrate a strong component. For example, some strong cryptographic algorithms are very difficult to break, it takes year before you do so, encrypted information are unlikely to attack communicated in a network.
- The Principle of Complete Mediation – In this case, a software system that requires access checks to an object each time an object 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, Email sender can be forged, Window "control+alt+delete", Secure interface, Input validation. Do not authenticate based on IP source, safe load, Hidden fields Safe login.
- The Principle of Least Common Mechanism – Granting access to a resources having multiple objects that share those mechanisms should be avoided. For example security activities can be add here to reduce potentially dangerous information flow, reduce possible interaction and make it more flexible.
- Principle of Economy of Mechanism - Complexity is one of the factor of evaluating a system's security. The likelihood that security vulnerabilities will exist within the system increases, if the design, implementation, or security mechanisms are highly complex. For example, for 39nalysing the source code that is responsible for normal execution of a functionality is a difficult task, but alternative behaviours checking in the remaining code may prove even more difficult in achieving the same functionality. Simplifying design or code is not always easy, but, when possible developers should strive for implementing simpler systems.
- The Principle of Reluctance to Trust – Assumption should be made by developers that the environment in which their system resides is insecure. Trust—whether it is extended to external systems, code, or people—should always be closely held and never loosely given. Software engineers should anticipate malformed input from unknown users when building an application. They are susceptible to social engineering attacks, even if users are known making them potential threats to a system. Similarly, no system is ever 100 percent secure, so the interface between two systems should be secured. The security of your application can be increase by minimizing the trust in other systems.
- The Principle of Never Assuming That Your Secrets Are Safe – Before an attacker launch an attack you should assume that attacker obtain enough information about your system. For instance, tools such as disassemblers and decompilers may allow attackers to obtain sensitive information that may be stored in binary files. Also, insider attacks, which may be accidental or malicious, can lead to security exploits. Sensitive information can be protected using real protection mechanisms as a means of protecting your secrets.
- The Principle of Psychological Acceptability – Security mechanisms should not be inhibited by accessible resources. If the usability or accessibility of resources is hinder by security mechanisms then users may opt to turn off those mechanisms. Security mechanisms should be transparent to the users of the system where possible or, at most, introduce minimal obstruction. In a software application security mechanisms should be user friendly to facilitate their use and understanding.

- The Principle of Promoting Privacy – Software systems can be protected from attackers who may obtain private information in an important part of software security. Customer may lose their confidence in the software if an attacker breaks into a software system and steals private information about a vendor's customers. This can be protected by preventing attackers from accessing private information or obscuring that information can alleviate the risk of information leakage.

3.1 Conventional Security Attacks

There are several attacks and software weaknesses that can cause damages to some important part of software. Below is the description of a few attacks that how these attacks happen.

- Cross site scripting (XSS): This is a form of injection vulnerabilities wherein, a malicious script is injected into a website. That script is injected into the dynamic content of the web site. It is then sent to the web user without any validation. Almost every technology or language used for website generation, like ASP.NET, ASP, CGI, JSP, Perl, C# and PHP, may encounter this vulnerability.
- SQL injection: When a dynamic SQL statement is built with user input, it allows an attacker to refine the statement's meaning and execute arbitrary SQL commands.
- Format string problem: With respect to this problem programming language C/C++ is the most vulnerable. It occurs when the user is able to control or write completely the format string used in printf() style family functions.
- Illegal Pointer Value: This problem is caused by a pointer pointing a location outside the buffer boundaries. Subsequent operations on this pointer could lead up to unpredicted results. This can be addressed at the implementation level, by addressing arrays instead of manipulating pointers should be used.
- Command injection: on behalf of an attacker, executing commands from an untrusted source cause an application to execute malicious commands. By command execution, the application gives an attacker a privilege or capability that the attacker would not otherwise have.
- Log forging: Attacker can hide their activities and disable tracking because of unauthorized modification of log files [6]. To avoid this problem at design level, an important point is to exclude untrusted source from logging.
- Path traversal: The best solution to this form of attack is input validation, if attacker can track, control the paths used in system file, he will be able to access protected system resources.
- Weak cryptography: One of the important software vulnerabilities is lack of data encryption especially sensitive information like authentication. Consideration of some guidelines like complex passwords during design phase may be useful. To prevent occurrences of this vulnerability select a proper encryption method which is one of the important decisions in implementation phase.

After knowing the importance of security principles and presence of several attacks, it is appropriate to look into the literature and find out *"what are approaches, or applications that address these security concerns using DSDM?"*

The table 1 presents the approaches, their limitation, and our observations about security concerns in those approaches.

Table 1. Presence of Software Security Concerns in Existing Agile Practices Including DSDM

Author	Issues	Observations	Software Security Principle
Bryan Sullivan, 2010	Traditional security development life cycle is suit for Agile. Proposes a new security development life cycle for agile[10].	Risk management activities unavailable. Security should be included in every phase, not from the beginning.	Unavailable
Kim, Y.-G. and Cha, S, 2012	Majority of system engineers do not have the relevant security knowledge about issues like security risk analysis, and security mechanisms and services[11].	Most of the system engineers lack the security knowledge.	Unavailable
Xioacheng Ge,2007	The security should be considered from the beginning of development process [30].	Integrate security process from the beginning.	Unavailable
DSDM public version 4.2 2006	The combination of people knowledge with tools and techniques such as prototyping and MoSCoW rules to achieve tight project delivery within the timeframe DSDM provides a flexible yet controlled process that can be used to deliver new systems[12].	The premise that most software project fail due to people concern rather than technology concern.	Unavailable
Addison Wesley,2003	The method of DSDM can be successful if all the core principles of DSDM is applied in project. Absence of any principle will endanger the whole basis of DSDM [13].	All the security requirement should be added to each phase.	Unavailable
Stevens, J.L.B., 2011	Majority of security engineers lack the systems-engineering background required to approach a security problem in general [14].	The security Engineers does not have Technical know about the security concern.	Unavailable
VTT Technical Research Centre of Finland,2003	In DSDM all changes during development should be reversible and development team members must have authority to make decisions in software development[15].	Every change can take place at any point and can be reverted.	Unavailable
H. Schmidt, 2010	Security requirements gathering allow putting together information about malicious part of the environment and subsequently facilitate decision on how security breaches can be nullified[16].	Security awareness must be put in place.	Mentioned
DSDM Consortium 2002-2006	The security activities is created and maintained and updated throughout the project life cycle [17].	Lifecycle includes security requirement in every phase, not at the beginning	Mentioned
Sipponen et al 2005	To integrate security into agile software development techniques , security methods should be adaptable and agile to operate in changing conditions[18].	Security techniques should be adaptable.	Unavailable
C. Alberts, J. Allen, R. Stoddard, 2011	There are some models that provide the needed support across the life cycle to measure software security[19].	Models are provided that support the security at all level.	Mentioned
DianxiangXu and Kendall Nygard2005	The most important part of the system must be considered in order to secure a complex system,other techniques that are formal may consider the threats with high security risks[20].	Threat analysis is the issue of discussion here.	Mentioned

E. Dubois and H. Mouratidis, 2010	The consideration of security since from the beginning of software development is recognised by Requirements Engineering community[21].	Early consideration of security from the scratch.	Mentioned
Mustafa k. et al 2008	The mathematical modelling of Security policies is one of the forefront research in the field of security. However, security policies are mostly considered at the SLDC[22].	At the first stage of software development.	Mentioned
Gunnar Peterson, Arctec Group 2006	The architecture of Software security architecture is an iterative process that decompose complex problem spaces , drilling down on granular details to gain traction in a domain; and then synthesizing across domains, building up design views, identifying relationship vectors that illustrate the system's security design goals[23].	Security to be split into smaller problem and dealt with accordingly.	Mentioned
P.Abrahamsson ,et al 2010	The incremental architecture and DSDM is effective in managing security requirement changes[24].	Security change can be managed in incremental architecture.	Unavailable
D. Mellado, E. Fernandez-Medina, and M. Piattini, 2010.	The iterative and incremental process can be integrated in organization's Software Product Lines development process model so that it provides security requirement engineering approach[25].	Security in agile is also iterative and incremental in nature.	Mentioned
H. Mouratidis, and J. Jurjens, 2010	Security can be implemented within system code either as security controls or as security attributes that need to be enforced on the application's operations [26].	Security should be enforce at the application stage.	Unavailable
B. Morin, T. Mouelhi, F. Fleurey, et al.2010	Majority of software security are limited in expressiveness, flexibility and software engineering process and tool support. However, adaptation approaches integrate less run time security configuration of components [27].	It lacks configuration of security component.	Unavailable.
A. Fuchs, S.G¨urgens, and C. Rudolph, 2011	Security modelling can be fulfilled by the system without modelling of attacker capabilities, but rather through security guarantees [28].	No security guaranteed.	Unavailable
M. Waterman, J. Noble, and G. Allan, 2012	Agile practitioners has found that developers use five broad strategies to determine their level of up-front architecture .However, the remaining four vary in their levels of upfront architecture [29].	The architecture of agile is mentioned here.	Unavailable

4. ANALYSIS

Based on our literature (Table 1), eight (8) articles mentioned the security principles in relation to agile development, whereas thirteen (13) articles did not mention the implementation of security principles. However, only one (1) forum [17] mentioned the security principles in relation to DSDM. On the other hand, we could find only 5 articles mentioned DSDM with/without security issues (Figure 2). We can analyse that there are many suggestions on integrating security aspects in agile methods, in general. There are perceptions from investigations that security aspect should be incorporated at the beginning of software development life cycle. This shows that there is a tremendous need for integrating security phases, sub-phases or practices into DSDM so
that this method could be used to develop secure software.

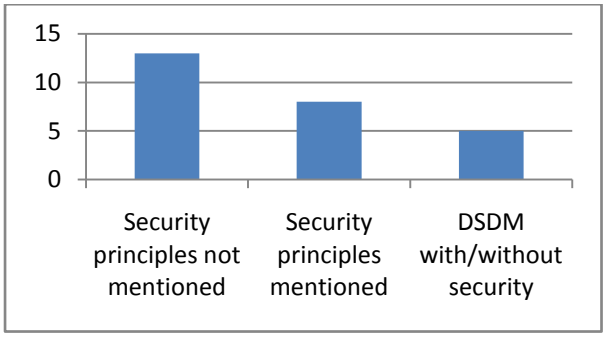

Fig 2. Security Principles in Agile Literature

From this analysis we can deduce that in DSDM security principles have not been considered.In Table 2, we provide our analysis about the techniques that implemented DSDM, in general with/without security.

Table 2.Limitations of Existing DSDM based Techniques in Relation to Security

Approaches	Security Phases	Security-Focused Roles	Security Attacks
Prototyping	√	X	x
Timeboxing	x	X	x
MoSCow prioritization	x	X	x
Facilitate workshops	x	X	x

Table 2 presents existing approaches that have adopted DSDM. However, only one (1) technique called Prototyping used additional security phases. While, most of other approaches did not include any security phases, security focus roles or security attacks in the DSDM. From this, we can see the gap that existing approaches do not significantly address the security issues of software while implementing DSDM.

5. CONCLUSION AND FUTURE WORK

Similar to other agile methods, DSDM is also gaining popularity because of its incremental and iterative features for software development. However, most of the researches focus to improve the general efficiency of DSDM. Based on our literature review, only one research discusses about integrating security into DSDM. Thus, there is a need for intensive research to study the suitability and adaptability of DSDM to cater the security problems of software. In this review paper, we analysed literature aboutDSDM whether it addresses software security or not. Based on our own analysis about the DSDM, we found that in its current form DSDM does not support secure software development. It means that if software is developed using DSDM then that software may not be secure software. The foremost reason is that, in the fundamental structure (life cycle) of DSDM security aspect of software development was ignored. Thus, we intend to enhance the existing DSDM structure so that the security problem can be reduced to minimum. Therefore, our future research plan is underway to refine existing DSDM into a secure DSDM. The results of our effort will be shared with the agile community and published in the coming events soon.

6. ACKNOWLEDGMENTS

We would like to express our gratitude to Ministry of Science, Technology and Innovation (MOSTI) Malaysia for funding this research project under Vot: 4S028.

7. REFERENCES

[1] Aydal E.G. 2005. Extreme programming and Refactoring for Building Secure Web based Applications and Web Services, MSc Project, University of York.

[2] M. E. Zurko and R. T. Simon 1996. User-centered security. In Proceedings of the 1996 workshop on New security paradigms, 1996, pp.27–33.

[3] CERT Statistics, [Online].Available: http://www.cert.org/stats/. [Accessed: 30- 04-2013]

[4] Saltzer, Jerome H. & Schroeder, Michael D.1975. The Protection of Information in Computer Systems, 1278-1308. In Proceedings of the IEEE

[5] Viega, John & McGraw, Gary. 2002. Building Secure Software: How to Avoid Security Problems the Right Way.Boston,MA:Addison-wesley.

[6] AntiVaha-Sipila. 2013. Marrying Scrum and Security blog, 19 May 2009. http://blog.safecode.org/?p=45.ht [Access on 30-04-2013]

[7] http://www.DSDM.org - the home site for the DSDM consortium.[Access on 30-04-2012].

[8] R.C.Martin. 2002 . AgileSoftwareDevelopment: Principles, Patterns and Practices', Prentice Hall (October 2002).

[9] Chivers H., Paige R.F., and Ge X. 2007. Agile Security Using an Incremental Security Architecture. Department of Computer Science, University of York.

[10] Bryan Sullivan 2010. Bryan Sullivan, Streamline Security Practices For Agile Development. MSDN Magazine.

[11] Kim, Y.-G. and Cha, S. 2012. Threat Scenario-based Security Risk Analysis using Use Case Modelling

in Information Systems. Security and Communication Networks, 5(3), pp.293-300.

[12] DSDM PublicVersion 4.2. http://www.dsdm.com/products/dsdm_version_4_2.asp Access [30.04.2013].

[13] Stapleton J. 2003. DSDM Business FocusDevelopment, Second Edition,Addison Wesley.

[14] Stevens, J.L.B. 2011. Systems Security Engineering. IEEE Security & Privacy, pp.72-74.

[15] Koskela, J. 2003. Software configuration management in agile methods. VTT Technical Research Centre of Finland.

[16] H. Schmidt, McDermott, J. & Fox, C. 1999. Using abuse case models for security requirements. In Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC). IEEE Computer Society.2010, pp.188 –195.

[17] DSDM Consortium 2002-2006 www.DSM.com.[Access on 30-04-2013]

[18] Sipponen, M., Baskerville, R. &Kuivalainen, T. 2005. Integrating security into agile development methods. In Proceedings of the38th Hawaii International Conference on System Sciences.

[19] C.Alberts,J.Allen,R.Stoddard. 2011. Risk-based measurement and analysis: Application to software security. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Technical note CMU/SEI-2011-TN-032.

[20] DianxiangXu and Kendall Nygard. 2005. A Threat Driven Approach to Modelling and Verifying Secure Software. In the Proceeding of the 20thIEEE International Conference on Automated Software Engineering, 2005.

[21] E. Dubois and H. Mouratidis. 2010. Guest Editorial: Security Requirements Engineering: Past, Present and Future. Requirements Engineering, vol.15, no. 1, pp. 1–5.

[22] Mustafa k. et al. 2008. Development of Security Assessment Framework for Object Oriented Software. Project Report, Submitted to DIT, Ministry of Communication and IT, Govt. of India.

[23] Security Architecture Service Oriented, Gunnar,Arctec Group 2006.

[24] P.Abrahamsson, M.Ali Babar, and P.kruchten, Agility.2010. Architecture :Can They coexist?. IEEE, Software.vol 27,no 2.

[25] D. Mellado, E. Fernández-Medina, and M. Piattini. 2010. Security requirements engineering framework for software product lines. Information and Software Technology vol 52:p.1094-1117.

[26] H. Schmidt. 2010. Threat- and risk-analysis during early security requirements engineering. In the Proc. of Availability, Reliability, and Security,ARES'10 International Conference, IEEE Computer Society.

[27] B. Morin, T. Mouelhi, F. Fleurey, et al. 2010. Security-driven model-based dynamic adaptation. In Proc of 25th IEEE/ACM Int.Conf.on Automated software engineering, Belgium, pp. 205-214.

[28] A. Fuchs, S. Gürgens, and C. Rudolph. 2011. A formal notion of trust and confidentiality – Enabling reasoning about system security. In Journal of Information Processing, vol. 19, pp. 274–291.

[29] M. Waterman, J. Noble, and G. Allan. 2012. How Much Architecture? Reducing the Up-Front Effort. In Agile India 2012, pp. 56–59.

[30] Xiaocheng Ge, Richard F. Paige, Fiona Polack 2007. Extreme Programming Security Practices. G.Concas et al. (Eds.): XP 207, LNCS 4536, pp. 226-230, 2007. Springer-Verlag Berlin Heidelberg 2007.

[31] Azham, Z., Ghani, I., Ithnin, N. 2011. Security Backlog. In Scrum Security Practices, 5th MySEC (Malaysian Conference in Software Engineering), 2011.

[32] Imran Ghani, Izzaty Yasin. 2013. Software Security Engineering in Extreme Programming Methodology: A Systematic Literature Review. Sci.Int.(Lahore),25(2),215-221.

[33] Adila Firdaus, Imran Ghani, Nor Izzaty Mohd Yasin. 2013. Developing Websites using Feature Driven Development: A Case Study. Journal of Clean Energy Technologies, Vol 1, No 4.