ISSN No. 0976-5697

Volume 3, No. 1, Jan-Feb 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Security Test Method on Agile Software Development Based on ISTQB

Hossein Janeh and Sam Jabbehdari
Department of Computer Engineering,
Islamic Azad University,
Tehran North Branch, Iran
hjaneh@gmail.com, sjabbehdari@gmail.com

Ramin Nassiri*
Department of Computer Engineering,
Islamic Azad University,
Tehran Central Branch, Iran
r_nasiri@iauctb.ac.ir

Abstract: Today, Web applications are growing rapidly. Considering Enterprise requirements to acquire unified systems and web-base applications, agile method(s) could be conceived as a major development method for web solutions at the first glance. Thus, Security is the most important non-functional requirement in a development process. In this paper, we present a method to test security requirements in agile software development, based on ISTQB, which is globally known as the recent robust framework for software testing. Deploying ISTQB would be present standardization to our proposed method.

Keywords*:* Agile development, Security testing, ISTQB, Web application.

I. INTRODUCTION

Agile methods are developed against the ad-hoc traditional methods such as waterfall in addition to a few recent ones [1].Ordinary methods are based on several steps that more often begin with requirement specification, Continued with design and followed by ordinary steps until at the end a software product is obtained. These traditional methods actually are suffered while there are urgent needs for change according to the dynamism in user environment [2]. Agile means nimbleness in change according to customer's frequent changing in requirements. Technical progress in developing web base system and moving to iterative process may get us the idea of agile security testing [3].

ISTQB is a world-wide non-profit organization responsible for defining various guidelines such as test structure and regulations, accreditation; certification, etc. Working groups within the ISTQB are in charge of developing and maintaining a variety of software test techniques in addition to the syllabi and exams provided to software tester training courses. The ISTQB is assisted by representatives from each existing National and Regional Board. A National Board is a working group of testing specialists from a specific country or region. Members actually are professionals and recognized experts from industry, consultants, trainers, academic professors, scientists and specialists from other organizations [4]. In ISTQB, there are four distinct levels for testing: 1.component testing 2.integration testing 3.system testing and 4.acceptance testing [5].

We map our method to these four levels of testing and employ ISTQB framework for testing security in each level. Then we use our method in a case study. The results show that our approach is suitable enough for security testing.

II. BACKGROUND AND RELATED WORKS

A. *Security in Agile Development:*

First, we use some definitions from ISTQB standard Glossary [6]:

a. ***Security****:* Attributes of software products that bear on its ability to prevent unauthorized Access, whether accidental or deliberate, to programs and data.
b. ***Security testing tool:*** A tool that provides support for testing security characteristics and vulnerabilities.
c. ***Security testing:*** Testing to determine the security of the software product.
d. ***Agile software development:*** A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. [6]

The Agile Manifesto's core principles and security has significant mismatches have been identified by various writers and panelists such as satisfy the customer unless customer is highly security-aware.[7]

In[8] XP method was analyzed from a security engineering standpoint. This is done by analyzing XP in the light of two security engineering standards; the Systems Security Engineering-Capability Maturity Model (SSE-CMM) and the Common Criteria (CC). The result is that XP is more aligned with security engineering than one might think at first. However, XP also needs to be tailored to better support and to more explicitly deal with security engineering issues.[8]

Another research has shown how the security features can be augmented into agile methods. Key security features are mentioned in all phases of development. They are Security-relevant subjects, Security-relevant objects, Security classification of objects and subjects and Risk management. First, security-relevant objects are identified. Then upon it's conduct, the security-relevant subjects are identified. After that then it would be possible to classify security-relevant objects and subject and risk analysis from it in requirement phase. In test phase, test selected features functionalities and qualities according to the essentials [9].

Keramati and et al [10] focused on agile methodologies in order to empower them with security activities. For each security item, first one should determine an agility degree. After that deploys security activity ordered by the high level agility degree to low agile activity and the minimum degree is the new mixed activity degree. If this degree is acceptable, then the security activity is passed, and may proceed to next security activity. This algorithm continues until all security activity is tested. This could be used for penetration testing which ended up with the last agile activity [10].

B. Securities Testing in Agile:

Agile Security Testing of Web-Based Systems via HTTPUnit is done by employing a highly testable architecture and using an automated testing framework. t The farmeworkcan bypass the presentation layers and interact directly with the underlying web application server via HTTPUnit. This gives the team the ability to perform security testing for critical vulnerabilities that are best mitigated by secure programming practices on the web application server. [3]

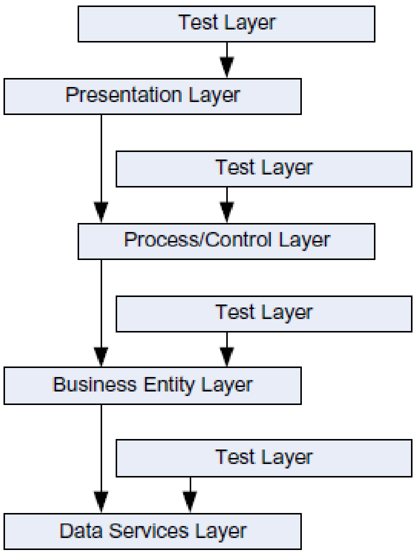

Figure 1. highly testable architecture as shown in [3]

In Figure1 one can see this architecture. This means that there is a software layer dedicated to testing the data services layer, another layer dedicated to testing the business services layer, and a third dedicated to testing the presentation layer. [3]

In" Security Testing in Agile" by Gencer Erdogan et al [11], AST (Agile Security Testing) method is extended, and three additional stages are augmented to it as : 1. Penetration testing and mitigation false positive: After each execution of penetration testing, reviews the result and detect false positive . This needs tools to mark false positive and preventing them in the next execution. 2. Postmortem evaluation: It shows the reason that why some bugs are not discovered in development phase and patching the test tools for cover them. 3. Knowledge repository: It is for saving some valuable information. The new method has named Extended Agile Security Testing (EAST) and includes a few steps as: 1. Design misuse case. 2. Use testable layer architecture.3.Automatic code review. 4. Fill knowledge repository.5.Penetration testing and mitigation false positive. 6. Postmortem evaluation [11].

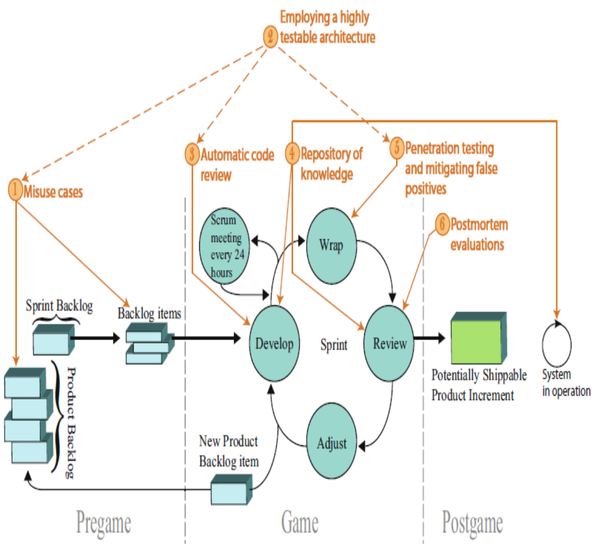

Figure 2. Extended AST method by [8]

Extending Agile Security Testing and integrating it into Scrum shown in figure 2.

III. THE PROPOSED METHOD

We need a feedback for our development in the security testing process, so we get the idea of weighting and make four arrays according to ISTQB four levels in testing software: component, integration, system and acceptance test. Figure3 shows the V- model in ISTQB[12].The four layers on the right side appearance. Definition of these layers in ISTQB:

- ***a)*** ***Component testing:*** The testing of individual software components.
- ***b)*** ***Integration testing:*** Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
- ***c)*** ***System testing:*** The process of testing an integrated system to verify that it meets specified requirements.
- ***d)*** ***Acceptance testing:*** Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

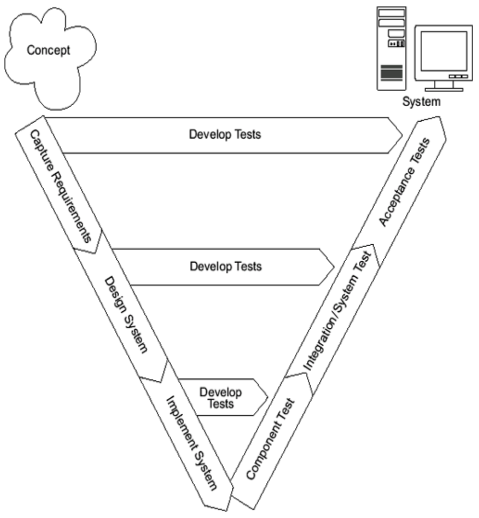

Figure 3. V-model in ISTQB[12]

We map each security item to ISTQB's layer test. and use ten items from Stephen de Vries article" Security Testing Web Applications throughout Automated Software Tests"[13] and replace acceptance testing with system testing, because in ISTQB, the concept of system testing adapts to it. The result is shown in table1.

Table 1. ISTQB and Security Items

Items	Component Testing	Integration Testing	System Testing	Acceptance Testing
SQL Injection		Static analysis	Use Case Testing	
Input Validation	Static analysis	Static analysis	Equivalence Partitioning	
Special Characters	Static analysis	Static analysis	Equivalence Partitioning	
Active script injection			Use Case Testing	
Authorization		Control flow	Use Case Testing	
Cookie Transport		Control flow	Use Case Testing	
Logout/Log off		State Transition	State Transition	
Expiration		State Transition	State Transition	
HTML Injection			Use Case Testing	
Lockout		State Transition	State Transition	

a. Each array has a number of test items we can test on those. For example, input validation is in the first three levels, and HTML Injection is in the system testing. These arrays are 2 dimensional, first dimension is the security item name and another shows the weight of this item. (Related to our work).this value is a number between 0 to 5.

b. After a first iteration, we have created a single element (such as a class) thus we refer to array one(component array) and choose a highest weight element for testing in our project. Duration time we spent on it depends on deliver planning to customer.

c. In the integration phase, we do similar work as we did in the component level. If a fault detects for an item and if that item exists in the previous array, which means this fault may not be discovered in previous step, so we add one unit to its weight in the previous array to emphasize it for another development. It may be increase the ability to find such defect in earlier steps in future. So the cost of finding and fixing defects will be fewer.Figure4 from [5] shows this clear.

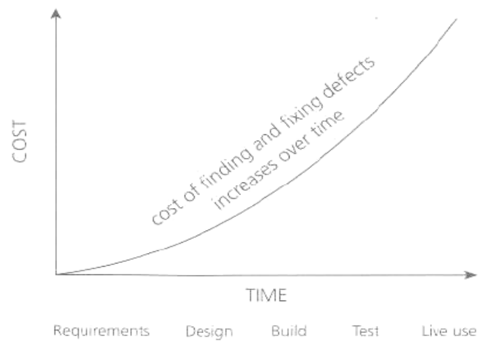

Figure 4. Cost of defects[5]

Sometime, it is possible that responsible team(s) for one component doesn't fulfill its job in the appropriate time so the component integrated with it cannot be tested. It is better to postpone testing than ignore test.

Figure 5 s that the security test place in extreme programming (XP) agile method. The dotted border includes the XP iteration phase. It continues until the final product is completed. Pair programming is in the Coding step, and the team may define a security tester and in this way one of the programmers writes the code and other does the security testing.

d. In the system testing, we have a whole system and in this step, we use ISTQB specification-based (Black box) techniques to discover the security bugs. If an item has a fault, and it is in the previous array (such as SQL injection), the one unit is added to its weight in the previous array.

We use new weights for next iteration.

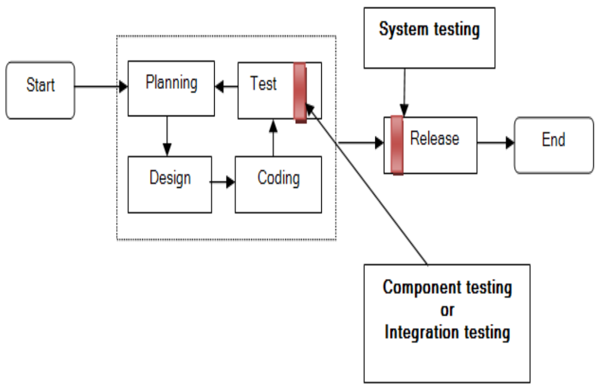

Figure 5. Security test place in extreme programming

We describe our algorithm in pseudo code:

```
Array C//this array have component testing element
//W(c) =weight of component c
Array S// this array have system testing element
Array I// this array have integration testing element
Switch (development phase)
Case “component test”
Do
{
Select item c from C where ∀c’∈ C, W (c’) ≤ W(c)
Test c;
Remove c from C;
}
Until (time expired or no item in C)
If (time expired && C is not empty)
For each c in(C ∪ U) increase W(c);
Case “integration test”
Do
```

```
{
Select item i from I where ∀i'∈ I, W (i') ≤ W (i)
Test i;
Remove i from I;
}
Until (time expired or no item in I)
If(time expired && I is not empty)
Foreach (i in (I ∪ S) increase W (i);
Foreach item in i
If(i has bug and i is in C) increase W(c) where i=c;
Case "system test"
Do
{
Select item s from S where ∀s'∈ I, W (s') ≤ W (s)
Test s;
Remove s from S;
}
Until (time expired or no item in S)
Foreach item in S
If(s has bug and s is in I) increase W (i) where s=i;
```

IV. CASE STUDY

We deployed this method in design and developing a helpdesk for interaction between employer and experts in systems. We used XP practice to accomplish it and each iteration approximately took an one and a half week. We developed our solution with MS visual studio 2010 and SQL server 2008. The outcomes are described hereafter.

a. ***Component Testing:*** in this phase, we use NUnit 2.5.10 for testing input validation and special characters, by writing suitable test cases which test classes for accepting usual input and despising banned ones. The same scenario used for testing special characters. NUnit is a free DotNet version of JUnit, a tool that consists of several functions and methods for testing single classes [14].

b. ***Integration Testing:*** in this phase, we use code analysis tools built-in visual studio.net 2010, and it can be use for static review our code and detect vulnerabilities. In some cases, it detects false positives. Code analysis has several security options and one of them is Review SQL queries for security vulnerabilities (CA2100), and we are confident that this item is enabled.

c. ***System Testing:*** the items in this phase use black-box techniques. For SQL injection, we use a misuse case scenario and with some tools such as FG-injector try to inject malicious code to our website. For input validation, use equivalence partitioning: we classified all possible input and select one member from each section.

d. ***Acceptance Testing:*** this level is related to the end-user, and we not suppose a special method for this.

We show how our proposed method works by figures depict by table2.

Table 2. Case study results

Iteration number	**duration**	**Test duration(including security test)**
Iteration1.	10 day	2 day
Iteration2.	11 day	3 day
Iteration3.	11 day	3 day
Iteration4.	10 day	2 day

Finally, we delivered our product to third party security tester before release, and the security degree of our software was good enough.

V. CONCLUSION

In this paper, we adapted our security testing, as a new method, to ISTQB and use ISTQB standard concepts as a general for security testing as a specific usage. We showed that ISTQB includes sufficient definitions and standards for our purpose, in addition to several benefits. First, this settles the time of test in our progress, and thus we have certain timelines for it. Second, the standard concept leads to a general concepts in all team's mind, and it prevents an excess time for matching team member together, especially when a new member added to team. Finally, It may push to Coding Standards that is one of XP's core practices.

VI. REFERENCE

[1] Vladan Devedˇzic´ and Saˇsa R. Milenkovi ,Teaching Agile Software Development: A Case Study ,IEEE,2010

[2] Victor Szalvay, co-founder, An Introduction to Agile Software Development, Danube Technologies, Inc,2004

[3] A. Tappenden, P. Beatty, J. Miller , A. Geras, M. Smith ,Agile Security Testing of Web-Based Systems via HTTPUnit, IEEE,2005

[4] http://www.ISTQB.org

[5] Dorothy Graham,Erik van Veenendaal ,Isabel Evans ,Rex Black, Foundations of Software Testing,2010

[6] Standard glossary of terms used in Software Testing, Version 2.1 (dd. April 1st, 2010)

[7] State-of-the-Art Report (SOAR) ,Software Security Assurance, July 31, 2007

[8] Jaana Wäyrynen, Marine Bodén, and Gustav Boström, Security Engineering and eXtreme Programming:An Impossible Marriage?, XP/Agile Universe 2004, LNCS 3134, pp. 117–128, 2004

[9] Mikko Siponena, Richard Baskervilleb and Tapio Kuivalainena , Integrating Security into Agile Development Methods, IEEE,2005

[10] Hossein Keramati, Seyed-Hassan Mirian-Hosseinabadi, Integrating Software Development Security Activities with Agile Methodologies, Sharif University of Technology , IEEE, 2008

[11] Gencer Erdogan, Per Håkon Meland and Derek Mathieson,"Security Testing in Agile web Application Development", Agile Processes in Software Engineering and Extreme Programming, Volume 48, Part 1, pp.14-28,Springer-Verlag Berlin Heidelberg.,2010

[12] Rex Black,Advanced Software Testing Vol. 1, Rocky Nook ,October 2008

[13] Stephen de Vries ,Security TestingWeb Applications throughout Automated Software Tests, Corsaire Ltd. 3 ,United Kingdom

[14] http://www.nunit.org