Towards Incorporation of Software Security Testing Framework in Software Development

Nor Hafeizah Hassan, Siti Rahayu Selamat, Shahrin Sahib and Burairah Hussin

Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka,
Durian Tunggal, Melaka,
Malaysia
{nor_hafeizah,sitirahayu,shahrinsahib,burairah}@utem.edu.my

Abstract. The aim of this paper is to provide secure software using security testing approach. The researchers have reviewed and analyzed the software testing frameworks and software security testing frameworks to efficiently incorporate both of them. Later, the researchers proposed to fully utilize the acceptance testing in software testing framework to achieve by incorporating it in software security testing framework. This incorporation is able to improve the security attribute needed during requirement stage of software development process. The advantage of acceptance test is to expose the system of the real situation, including vulnerability, risk, impacts and the intruders which provide a various set of security attribute to the requirement stage. This finding is recommended to establish a baseline in formulating the test pattern to achieve effective test priority.

Keywords: test pattern; software testing frameworks; security testing framework; security requirement; software process.

1 Introduction

In software industry, the testing process plays a crucial role, as people try to find defects of software [1]. The defects reflect the quality status of software on whether it should be ready to release. According to Standish's CHAOS Summary 2009, only 32% software was released successfully and others were neglected for many reasons [2]. Among the reasons for the negligence is the existence of defects in the software and thus, it is unfit to be deployed.

However, in the last few years, the industry is swamped with the term security testing which is claimed to find vulnerabilities in software [3]. The vulnerabilities are affirmed as any flaw that may be exploited by a threat. A research by US-Computer Emergency Readiness Team (CERT) shows total of 6000 vulnerabilities was cataloged in the first three quarters of 2008 [2].

Therefore, to control the situation, many researchers develop their own models or frameworks to reduce the numbers of bugs and vulnerabilities as in [4] and [5]. This paper analyzed two perspectives of testing (software testing and software security testing) to analyze and propose the possible incorporation between the two in software development process or also known as software development life cycle.

The rest of the paper is organizes as follows: section 2 present related work of testing and security testing. Section 3 discusses the analysis approach used to evaluate the software testing and software security testing frameworks and the last two sections present discussion and future work.

2 Related Work

Testing is a task in software development process. However, there are various software development process style such as waterfall, V-shape, Rational Unified Process (RUP), agile, and secure software development process as discussed in [6], [7] and [8]. Therefore, the way testing is conducted within each style may vary. A comparison analysis of software development process style is conducted to determine the testing roles.

2.1 An Overview of Software Development Process

Software development process is a life cycle of developing software. A traditional lifecycle consist of processes, namely as: requirement, analysis, design, coding, testing and operation [6]. A modified lifecycle consist of business modeling process (as in RUP), timeline and iteration (as in V-shape, spiral, RUP and agile). In this paper, the process is also referred as stage to emphasis the element of sequence. A fundamental stage of software development process was grouped based on similar activities on both traditional and modified lifecycle as represented in Table 1.

Table 1. Summary of software development model.
(√ = the respective stage exist is the software development style, Not stated = the respective stage is not described or not exist in the software development style)

Stage	SOFTWARE DEVELOPMENT					
	Waterfall	Spiral	V-shape	RUP	Agile	Secure software development
Feasibility	Not stated	√	Not stated	√	Not stated	Not stated
Requirement	√	√	√	√	√	√
Analysis	√	√	√	√	√	√
Design	√	√	√	√	√	√
Coding	√	√	√	√	√	√
Testing	√	√	√	√	√	√
Operation	√	√	√	√	√	√

As shown in Table 1, the Waterfall, V-shape, Agile and Secure Software Development consists of six stages starting from the requirement stage. On the contrary, the Spiral and RUP consist of seven stages including the feasibility stage.

Each of the stage has unique roles with their own set of activities. Feasibility is a stage of understanding the business concept; determine the objectives, alternatives and constraints; and estimate monetary resources. Requirement is a stage of elicit the stakeholders requirements within the project scope for what the system must do. Analysis is a stage of breaking down the requirement into workable process. Design is a stage of transforming the analysis into its architectural view, which may include high level design and low level design. Coding is a stage of writing the programming code into modules, test the individual developed modules (unit test), and integrate with other author modules as a functional system. Testing is a stage of checking the code implementation which covers various levels such as unit test, integration test and acceptance test with the objective to find defects, fixed it, and later verify it with the requirement. Operation is a stage of deploying the user accepted system into real environment, provide training, get it exploited, receive feedback and perform maintenance. The activities in software development stages are summarized as in Fig. 1.

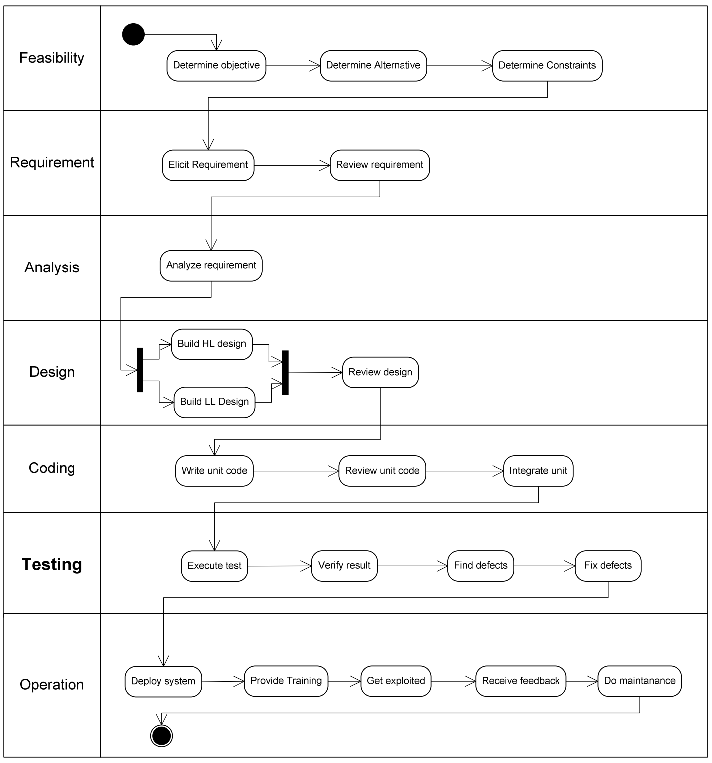

Fig. 1. Activities in software development stages. Testing is a stage of checking the code implementation which covers various levels such as unit test, integration test and acceptance test with the objective to find defects, fixed it, and later verify it with the requirement.

Thus, given a typical or a secure software development process model, the testing is a stage between coding and operation as shown in Fig. 1, and any dissatisfaction of testing stage require an assessment either at coding stage, design stage or requirement stage. To further understand the testing stage, an overview of software testing component is discussed in the next section.

2.2 An overview of Software Testing

Software testing (ST) is a process of executing a program to find the defects. Testing involves searching for runtime failures and recording information about runtime faults [9].

There are two basic components in ST. First is the test strategy which explains the way a test is conducted; involve the reveal of the code. If the code is reveal, it is white-box testing, else, it is black-box testing. If the code is used to design a test for black-box testing, it is called grey-box testing. Second is the test level which address the entire software development life cycle and each test level is a foundation of the higher level. The three levels are unit testing, integration testing and acceptance testing. Each level is viewed by programmers, designers, and business users respectively [10]. Both, the test levels and test strategy are summarized as shown in Fig. 2.

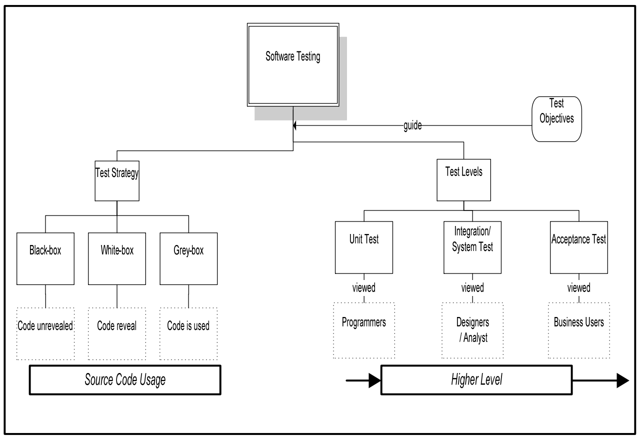

Fig. 2. The basic components in software testing: a) test strategy consist of black-box, white-box and grey-box, b) test level consist of unit test, integration or system test and c) acceptance test.

The test strategy assists in accomplishing the test levels objectives. A unit test use white-box testing strategy, integration test use grey-box testing strategy, and both integration and acceptance test use black-box testing strategy [4]. The test strategy and test levels chosen are guided by the test objectives, i.e. to answer why testers do the test. Test objectives are the factors that classify a test into a specific test type as described in [10] and [11]. The test level is viewed either by (involved) programmer, designer or business developers. In short, basic testing components are test level, test strategy and users.

In order to show the importance of the testing stage, we highlight the expectation of test: a) to find defect, b) fixed it and c) verify with requirement. Therefore, a comparison of actual test result and expected test result must be conducted. Any discrepancy of the two shall trigger the tester to trace the origin of error or fault.

The issue was pointed out as early as in 1970, when [6] proposed the manageable waterfall software development by introducing skip-over the code and analysis stages to design stage if the testing stage fail to satisfy the external constraints. In 1986, [12] imposed the risk, prototype, review and evolutionary elements in spiral software development. In their model, testing is a notion in more composite manner; known as the test level. The test level describes the incremental of test consists of unit test, integration test, and acceptance test. However, there is no clear guideline of where to point back in case the testing result is not satisfied. It give the impression that coding and design, which are the nearest adjacent stages to unit testing, are good candidates for the purpose.

The same notion of *test level* is followed in V-shape model with illustration that each test level shall correspond to the development process stages respectively [13]. RUP, the IBM popular software development, denote testing as the verification process of all objects, integration and application requirements. Therefore, any dissatisfaction in testing caused a revisit to the code and requirements as concluded in [10] and [13]. In the iterative, incremental and user-centered agile software development, automated acceptance testing is emphasized to keep the prototype evolve in the next iteration [8]. The relatively new secure software development, proposed a secure development process within four stages, namely as: requirement, design, code and feedback [11]. The design stage consists of analysis and design, where as the feedback stage is the deployment stage. Major test is done in code stage such as risk-based security test and pent-test. Another secure software development by Microsoft, combined both spiral and waterfall approach, outlined an extended stage of testing named as verification stage prior to release and response stage [14]. Again, it is not clearly mentioned here, at which stage shall the developers return to in the event of test discrepancy occurs. Based on the stage sequence and the artifacts produced, the assumption is to check the code, design or requirement stage.

Currently, the aim of ST is to find errors or to verify test result with requirements [15]. However, the complexity of software had emerged and invites unintended users to penetrate the system. Hence, testing for boundary values only is insufficient to guarantee system functionality and minimize potential vulnerabilities. As a result, security testing is required to overcome this current issue.

2.3 Software Security Testing

The current practice of software security testing (SST) is to detect any defects that contribute to the flaws exhibit in an application and always considered as an afterthought concerns [11]. SST is concerns with two objectives: first, the test is executes to find what should not happened within a system and second, is to disclose any attack from intruders. The attack can breach the security by exploiting the vulnerabilities (weakness) exist in the application. Therefore, the software security testing scope requires a tester to be equipped not only with testing expertise but also with software security knowledge [11].

The distinction between ST and SST come in twofold. Firstly is on the second objective of the SST, i.e. the existence of intruder or attacker element as discovered by [1] and [16]. Secondly is on the existing research of ST which focuses on how to combine the testing components to produce an effective testing result, where as SST focus on how to eliminate the afterthought concern by incorporate the security aspect into the whole software development from the beginning stage. It is debatable that ST utilizes all the test components effectively and how SST could utilizes ST components to achieve its objectives [16]. To the best of our knowledge, the position of this SST within the current view on software testing diagram or vice versa is yet to be determined. Thus, an analysis on software security testing frameworks needs to be conducted in order to establish a comprehensive understanding of the SST within the software development.

3 Analysis Approach

In order to obtain the perspective, an analysis of software testing and software security testing frameworks is conducted

3.1 Software Testing Framework

Software Testing Framework (henceforth, STF) is an approach used to perform testing as effective as it can [5]and [10],. As the testing phase of a software life cycle is extremely cost intensive (40% of the whole budget) many researchers look into it with various perspectives. A framework could consist of an approach, a technique or a model. Generally, a testing framework execute using five steps: a) identify the test objective, b) generate testing input using application's specification, c) produce expected output results, d) execute and validate the test cases, and e) amend the application following with regression test [17] and [18]. The objectives of the test shall consider the type of application (software) under test. For example, during a test in web application software, the number of distributed users is enormous, an aspect that need to be concerned.

In [4], they introduced meta programming in testing framework in order to reduce the test preparation load by overcome the three challenges in testing i.e. poorly design test cases, manual works and discrepant tools. They implemented the framework in unit test and system test level. The tool was developed using Java. Meanwhile, [18] and [19], proposed a multi-agent system architecture for automated testing framework in distributed environment to deal with different type of users. The framework in [19] focused on four properties during integration testing level, namely: interoperability, compatibility, function, and performance. [20] proposed a new algorithm to verify completeness and consistency property in web services . In an extended work of unit testing framework, Diffut, run on Java to test the difference between two same inputs using Rostra and Symtra [21]. A slight different domain, in electronic health records environment, introduce the Archetype Definition Language (ADL) in their testing framework [22]. Next, in [23], an improve framework of regression test is demonstrated in database application. The framework used DOT-select as it test tool which is part of a larger Data-Oriented-Testing framework that is under development at the University of Manchester. In another work, [24] explained the testing framework for model transformation, that is to test the changes in graphical model. The framework used an extended declarative language, Embedded Constraint Language (ECL) for describing rules (applied in UML). [25] introduced a web services test framework by mapping the Web Service Definition Language (WSDL) to Testing and Test Control Notation Version 3 (TTCN-3) using a third party test tool, TTworkbench. TTworkbench is the full-featured integrated test development and execution environment (IDE) test automation.

The frameworks are selected based on as view by (involve) developer, such as unit test or integration test or regression test. In order to review the framework, the common fundamental aspects in software development are extracted. This research discloses that the aspects used to determine the perspective in software testing framework are programming language, tools, domain, standard, test level and test strategy as summarized in Table 2.

Table 2. Common aspect extracted from software testing framework.
(Not stated = the common aspect is not found in the reviewed framework*)*

Author	**Common Aspect**					
	Language	**Tools**	**Domain**	**Standard**	**Test level**	**Test strategy**
[4]	Java	JUnit, CUnit, CPPUnit	Distributed	Not stated	Unit test	White-box
[18]	Object-oriented	Not stated	Web-based application	Not stated	Unit test Integration test System test	White-box, Grey-box
[19]	Java	JADE	Distributed	Unified interface standard	Integration test	White-box, Grey-box
[20]	OWL-S	Not stated	Web-based application	SOAP	Unit test, Integration test	White-box, Black-box
[21]	Java	Rostra, Symtra	Not stated	Not stated	Unit test	White-box
[22]	Archetype Definition Language	Not stated	Distributed Electronic Health Record	Not stated	Unit test	Black-box
[23]	XML	JDBC (DOT-select)	Database	Not stated	Regression test	Black-box
[24]	Embedded Constraint Language	Generic Modeling Environment	Model transformation	Not stated	Unit test, Integration test	White-box, Grey-box
[25]	WSDL	TTWorkbench	Distributed, Web-based application	TTCN-3, ETSI, ITU, SOAP	Unit test	Black-box

JADE=Java Agent DEvelopment Framework, OWL-S=Ontology Web Language for Service, WDSL=Web Service Definition Language, TTCN-3=Testing and Test Control Notation Version 3, ETSI=European Telecommunications Standards Institute, ITU=International Telecommunication Union , SOAP=Simple Object Access Protocol

Based on Table 2, the programming language aspect explains any specific language used in the framework, either as object-oriented or specific name such as

Java, eXtended Markup Language (XML), ADL, ECL or WSDL. The tools aspect is any assistant used to demonstrate the author's chosen algorithm or techniques. The domain aspect focuses the platform of the software development such as in general (distributed or web-based environment) or specific (model transform, database or others). The standard aspect composed of any policy adhere and enforced by the author, such as access protocol in Simple Object Access Protocol (SOAP) TTCN-3or unified interface standard for open source. For example, most of web-based or distributed testing frameworks follow the standard used specifically in communication. Regression test, a re-test after fixing a bug is noted significantly in database application to maintain the database state whenever changes occurred. The grey-box test is relevant in two scenarios, firstly is when the code was not directly revealed such as in agent-based testing framework [18], [19] and model testing framework [24], but rather is invoked by other metadata. Interestingly, the methodology used such as waterfall or agile is not a major concern in these reviewed STF.

The findings of the analysis is then summarized and mapped into the software development stages as illustrated in Fig.3. Fig. 3 shows that test strategy is used to support the unit test and integration test. It is noted that the test components summarized in the testing framework (domain, language, tools and standards) limits the test level conducted by only developer and designer (based on unit test, integration test or regression test). These internal testers are claimed to assess the design stage to compare the test result [10]. Consequently, this situation allowed the review process to go as further as to design stage only.

Business user which is claimed to prefer review the requirement stage (as have minimum knowledge in software design or programming language) in finding out the discrepancy of test [13], is not yet engaged. The lack of business user involvement in the reviewed frameworks had limit utilization of test level to system test. Another test level, the acceptance test, only could be utilized in the existence of business users. The diamond shapes in Fig.3 denote the defect-by-developer is defect found as test conducted by developer and defect-by-designer is defect found as test conducted by designer.

Hence, the aspects discussed in this software testing framework are addition to existing testing components discussed in Section 2.2. Therefore, apart from test strategy and test level, a testing framework also depends on language and domain aspects assist by tools and benchmark by standards.

Based on the findings, this research proposed the issues tackled in the STF framework are simplified into five objectives : a) to achieve the objective of test (checking for completeness, consistency, interoperability as in [20], [22] and [24], b) to optimize the test as in database state [23], c) to reduce the test load as in [4] and [21], and d) to adhere the required test guideline (standard) as in [19], [20] and [25]. In order to add the essential security issue, an analysis of software security testing framework is discussed in the next section.

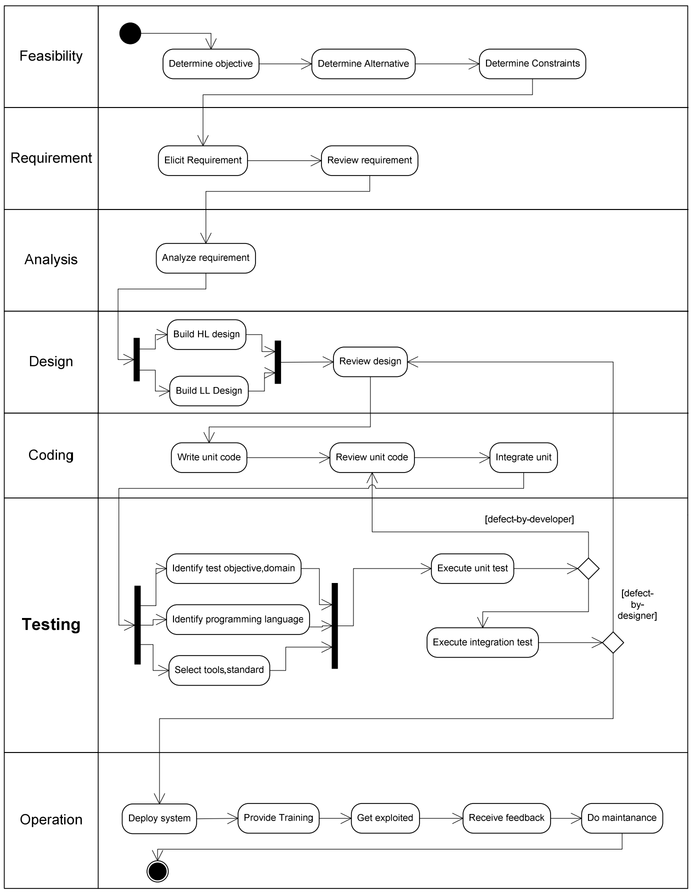

Fig. 3. The mapping of software development stages. The test components summarized in the testing framework (domain, language, tools and standards) limits the *test level* conducted by only developer and designer (based on unit test and integration test).

3.2 Software Security Testing Framework

Security testing framework (SSTF) is an approach used to test the security aspects of a product [16]. Always, this framework is considered as an afterthought concerns, i.e. the process only begin once the product is ready.

A comparison of security framework had been made by [26]. They compared eleven frameworks from year range 1996 to 2004. They highlighted the needs for a standardized methodological approach that taking into account security aspects from the earliest stages of development till the completion. Another work was conducted by [27] to summarize the security dimensions, such as cause, impact, and location, encountered in security frameworks. However, there is still lack of research done of how to integrate testing operation in software development process [28].

For the purpose of this paper, we examined eight SSTF frameworks to disclose the attributes involved. The frameworks are selected based on the security aspects integrate or enforced within. The selected SSTF are *Knowledge Acquisition Automatic Specification (KAOS), Model Driven Security (MDS), i*, Secure Tropos,* Security *Quality Requirements Engineering (SQUARE)* methodology, *Security Requirement Engineering Process (SREP), Security Requirement Engineering (SRE)* and *Threat Modeling*. *KAOS* started from cooperation between the University of Oregon and the University of Louvain (Belgium) in 1990. *KAOS* is a goal-oriented software requirements capturing approach in requirements engineering which has been extended to capture security threat using it anti-goals [29]. *MDS* is a framework that automatically constructing secure, complex, distributed, applications with the UML integration [30]. The *i** framework was developed for modeling and reasoning about organizational environments and their information systems which later embed the trust model [31]. *Secure Tropos* is an extended approach from *Tropos*. It explains a formal framework to model and analyze security requirements that focus on ownership, trust and delegation [32]. *SQUARE*, is a nine-step approach to elicit, categorize and priority security requirement [33]. *SREP* which is similar to *SQUARE* imposed the standards and policy enforcement (Common Criteria) within the framework [34]. *SRE* is a framework to determine how adequate is a security requirements [35]. There are particular frameworks that had been adapted into tools such as *Threat Modeling* by Microsoft. The findings revealed that those frameworks start their security consideration as early as requirement stage as depicted in Table 3.

Table 3. The stages focused in software secrurity testing framework.
(√= the software development stage that the model focus on, *Not stated* = the model is not clearly stated focus in this stage)

Software security testing framework	Stage	
	Requirement	Design
KAOS (Security Extension	✓	✓
MDS (Model Driven Security)	*Not stated*	✓
*i** framework	✓	*Not stated*
ST (Secure Tropos)	✓	*Not stated*
SQUARE	✓	✓
SREP	✓	✓
SRE	✓	✓
TM (Threat Modeling)	✓	✓

As shown in Table 3, we scope our analysis into both requirement and design stages for two reasons. First, according on a number of research [11], to have an efficient cost, testing should start as early as possible in product development. Second, to produce a general framework and suitable in any domain, an early stage is prominence. Table 3 illustrates that in SSFT, security is a concern as early as during the requirement stage. However, the requirement elicitation activities need a guideline to present the software security vulnerabilities effectively. Software security vulnerabilities are caused by defective specification, design, and implementation. Unfortunately, common development practices leave software with much vulnerability. In order to have a secure cyber infrastructure, the supporting software must contain few, if any, vulnerabilities. This requires that software be built to sound security requirements. Therefore, we propose the utilization of ST activities into SSTF to provide the attributes needed in constructing security requirements. The details are discussed in the next section.

4. Discussion and Findings

This research shows that current analysis focuses three factors. First, the affected stage - the STF concerns with on internal users and has tendency to review the test discrepancy at coding or design stage via unit or integration test. On the other hand, the SSTF required knowledge of security attribute to elicit its requirement as early as at requirement stage. Second, the collection of security attribute - the knowledge of security attribute is acquired based on SSTF objectives; to find what should not happen in a system and to reveal any attack from intruders. These attributes sources are best collected during the test level activities (unit test, integration test and acceptance test). However, the reviewed STF limits the test conducted during unit and integration or system test only. Consequently, the actual result does not reflect the whole test levels carried out within a system. Third, the issues of complex system - the issue of emerged complex system suggest that a system to be secure. On the other hand, the current STF cover at least five objectives except for security (see section 3.1). Hence, there is a need for software security testing within the software development stage. As a result, in this paper, we recommend to utilize the acceptance test within the software development stage by integrating it with relevance unit and integration test result. This integration shall provide a comprehensive test result to support the testing process as early as from the requirement stage.

4.1 The Proposed Framework

Based on the findings, the three factors discussed previously are incorporate into the software development stages to formulate a generic SSTF as illustrated in Fig. 4. Fig. 4 describes a testing process is guided by a test objectives, which denoted by attributes such as completeness, interoperability and compatibility. During the testing, all type of test strategy (black box and white box or both) is conducted depending on domain and language used in code. The testing process is executing manually or automatable assisted by tools. The actual output is compared with the expected output derived from design specification [10]. Any comparison discrepancy is returned during the feedback stage as review process. It is noted that the derived expected output is bound to the design process; hence, any inappropriateness shall lead to an inappropriate comparison at the feedback stage. Referred in Fig.2, ST activities involve programmer, designer and business user. Any test levels that involve the authors as the tester, the comparison is done alike – tracing the expected output from design stage. The acceptance test which involves business user and actual environment reviewed its test discrepancy between requirement stage and testing stage.

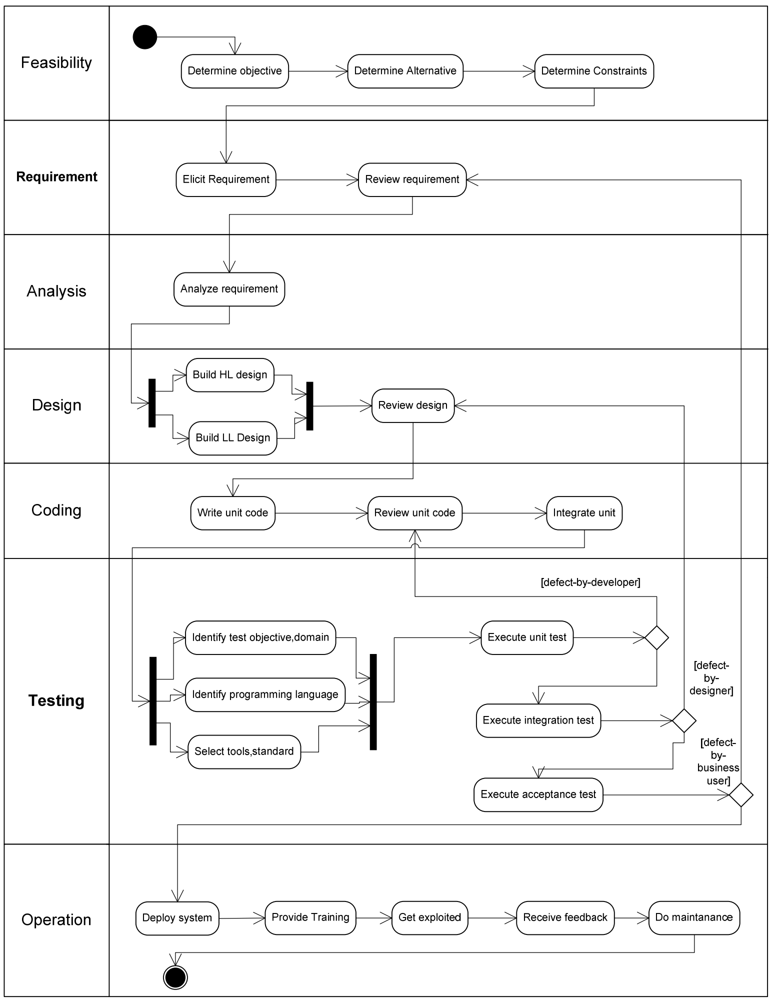

Fig. 4: Software Security Testing Framework. Any comparison discrepancy is returned during the feedback stage as review process. It is noted that the derived expected output is bound to the design process; hence, any inappropriateness shall lead to an inappropriate comparison at the feedback stage.

This research proposes the incorporation of the software security testing framework in software development by utilizing the software testing activities. The analysis proves that the software testing activities which focus on internal user (developer and designer) as tester has specific limit to overcome the test result discrepancy. Furthermore, the actual environment contributes to the existence of possible intruders which is important in security testing. As a result, the acceptance test which utilizes the business user and the actual environment is recommended to assist the test result discrepancy assessment. In other words, to preserve the test result consistency and completeness, the test result is best compared with the requirement stage.

In addition, the proposed framework consists of all stages in software development that are, feasibility, requirement, analysis, design, coding, testing and operation as summarized in Table 2. Therefore, the proposed framework can be used as a generic framework for security testing in any software development life cycle.

5. Conclusion and future work

In this paper, the software testing frameworks and software security testing frameworks are reviewed and analyzed. Based on the findings, this research proposed to fully utilize the acceptance testing in STF by incorporating it in SSTF. This incorporation is able to improve the security attribute needed during requirement stage of secure software development process. The acceptance test which exposed the system to real situation, including vulnerability, risk, impacts and the intruders shall provide a various set of security attribute to the requirement stage. Further improvement should be done in identifying the security attributes during acceptance test to generate a test pattern. This test pattern will further assist as a possible source in eliciting the security requirement during the requirement stage of software development. In addition, the proposed framework is expected to assist in tracing the web security attacks via a specific case study to generate a relevance testing pattern. The traceability process can be adapted in any other domain, such as in digital forensic investigation during collection of previous incidents data.

Acknowledgments. This research was kindly supported by Universiti Teknikal Malaysia Melaka under Short Grant Funding (PJP / 2009 / FTMK / (7E) S556) and Ministry of Higher Education Malaysia.

References

1. Thompson, H.H.: Why Security Testing Is Hard. J. Security & Privacy, IEEE. 1(4), 83--86 (2003)
2. Venter, H.S., Eloff, J.H.P, Li, Y.L.: Standardising Vulnerability Categories. J. Computers & Security, vol. 27(3-4), 71--83 (2008)
3. Jiwnani, K., Zelkowitz, M.: Maintaining Software With A Security Perspective. In: International Conference on Software Maintenance, pp.194--203 (2002)
4. Cho, H.: Using Metaprogramming to Implement a Testing Framework. In: ACM SouthEast Regional Conference. ACM, USA (2009)
5. Misra, S.: An Empirical Framework For Choosing An Effective Testing Technique For Software Test Process Management. J. Information Technology Management, vol. 16(4), 19--26 (2005)
6. Royce, W.W.: Managing The Development of Large Software Systems. In: IEEE Western Electronic Show and Convention, pp.1--9 (1970)

7. Rational Unified Process: Best Practices for Software Development Teams. Rational Software White Paper (2001)
8. Boehm, B., Brown, W., Turner, R.: Spiral Development Of Software-Intensive Systems Of Systems. In: 27th International Conference of Software Engineering, (2005)
9. Ko, A.J., Myers, B.A.: A Framework And Methodology For Studying The Causes Of Software Errors In Programming Systems. J. Visual Languages & Computing, vol.16(1-2), 41--84 (2005)
10. K.Mustafa, Khan, R.A.: Software Testing: Concepts and Practices. Alpha Science (2007)
11. Potter, B., McGraw, G.: Software Security Testing. J. Security & Privacy, vol.2(5),81--85, IEEE (2004)
12. Boehm, B.: A Spiral Model of Software Development and Enhancement. In: ACM SIGSOFT Software Engineering Notes, vol. 11(4), pp. 14--24, ACM New York (1986)
13. Craig, R.D., Jaskiel, S.P.: Systematic Software Testing. Artech House Publishers (2002)
14. Microsoft Security Development Lifecycle (SDL) Version 5.0, M. Library, Microsoft, http://msdn.microsoft.com/en-us/library/cc307748.aspx
15. Myers, G.J.: The Art Of Software Testing, New York: Wiley (1979)
16. Tondel, I.A., Jaatun, M.G., Jensen, J.: Learning from Software Security Testing. In: 8th IEEE International Conference on Software Testing Verification and Validation Workshop, pp. 286--294, IEEE Computer Society, Washington (2008)
17. Pu-Lin, Y., Jin-Cherng, L.: Toward Precise Measurements Using Software Normalization. In: Proceedings Of The 21st International Conference On Software Engineering, pp.736--737, ACM, Los Angeles, California, United States (1999)
18. Xu, L., Xu, B.: A Framework for Web Application Testing. In: International Conference on Cyberworlds, pp. 300--305, IEEE Computer Society, Washington (2004)
19. Jing, G., Yuqing, L.: Agent-based Distributed Automated Testing Executing Framework. In: International Conference on Computational Intelligence and Software Engineering, pp. 1--5, IEEE Press, Wuhan (2009)
20. Tsai, W.T., Wei, X., Chen, Y., Paul, R.: A Robust Testing Framework for Verifying Web Services by Completeness and Consistency Analysis. In: Proceedings of the IEEE International Workshop, pp. 159--166, IEEE Computer Society, Washington (2005)
21. Xie, T., Taneja, K., Kale, S., Marinov, D.: Towards a Framework for Differential Unit Testing of Object-Oriented Programs. In: 2nd International Workshop on Automation of Software Test. IEEE Computer Society, Minneapolis, USA (2007)
22. Chen, R., Garde, S., Beale, T., Nystrom, M., Karlsson, D., Klein, G.O, Ahlfedlt, H.: An Archetype-based Testing Framework. In: J. Studies in Health Technology and Informatic, vol. 136, 401-- 406 (2008)
23. Tang, J. , Lo E.: A Lightweight Framework For Testing Database Applications. In: Symposium on Applied Computing, ACM, New Zealand (2010)
24. Lin, Y., Zhang J., Gray J.: A Testing Framework for Model Transformations. In: Model-Driven Software Development - Research and Practice in Software Engineering, pp. 219--236, Springer, Heidelberg (2005)
25. Werner, E., Grabowski, J., Troschutz, S., Zeiss, B.: A TTCN-3-based Web Service Test Framework. In: Software Engineering Workshops, pp. 375--382 (2008)
26. Villarroel, R., Fernández-Medina, E., Piattini, M.: Secure Information Systems Development - A Survey And Comparison. J. Computers & Security, vol. 24(4), 308--321 (2005)
27. Igure, V.M. , Williams, R. D.: Taxonomies of Attacks and Vulnerabilities in Computer Systems. In: J. IEEE Communication Surveys & Tutorials, vol. 10(1), 6--19 (2008)
28. Maatta, J., Harkonen, J., Jokinen, T., Mottonen, M., Belt, P., Muhos, M., Haapasalo, H.: Managing Testing Activities In Telecommunications: A Case Study. J. Eng. Technol. Manage., vol. 26, 73--96, Elsevier Science Publisher, Amsterdam (2009)

29. Lamsweerde, A.v., Brohez, S., Landtsheer, R.D., Janssens, D.: From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In: Requirements for High Assurance Systems, pp. 49--56 (2003)
30. Basin, D., Doser, J., Lodderstedt, T.: Model Driven Security: From UML Models To Access Control Infrastructures. In: ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 15(1), pp. 39--91 (2006)
31. Yu, E., Liu, L.: Modelling Trust In The i* Strategic Actors Framework. In: Proceedings of the 3rd Workshop on Deception, Fraud and Trust in Agent Societies, LNCS, pp. 175--194, Springer-Verlag London, UK (2001)
32. Giorgini, P., Massacci, F., Mylopoulus, J., Zannone, N. : Modeling Security Requirements Through Ownership, Permission And Delegation. In: 13th IEEE International Conference on Requirements Engineering Proceedings, pp. 167--176, IEEE Computer Society, USA (2005)
33. Mead, N.R., Stehney, T.: Security Quality Requirements Engineering (SQUARE) Methodology. In: Proceedings of the 2005 Workshop On Software Engineering For Secure Systems- Building Trustworthy Applications, pp.1--7, ACM, New York (2005)
34. Mellado, D., Fernández-Medina, E., Piattini, M.: A Common Criteria Based Security Requirements Engineering Process For The Development Of Secure Information Systems. In: Computer Standards & Interfaces, vol.29(2), pp. 244--253 (2007)
35. Haley, C.B., R. Laney, and J.D. Moffett, Security Requirements Engineering: A Framework for Representation and Analysis. IEEE Transactions On Software Engineering, 34(1): pp. 133--155 (2008)