International Journal of Information Technology and Knowledge Management
July-December 2009, Volume 2, No. 2, pp. 257-261

REPLACEMENT OF S/W INSPECTION WITH S/W TESTING

Mitu Kumari[*], Archana Sharma[**] & Vipin Kamboj[***]

The goal of software inspection and software testing is to reduce the expected cost of software failure over the life of the software product. In the case of inspections, the defect trigger is defined as a set of values that associate the skills of the inspector with the discovered defect. Similarly, for test scenarios, the defect trigger values embody the deferring strategies being used in creating these scenarios. Software inspection is the generic name for a set of cost-effective ways of evaluating user interfaces to find usability problems. They are fairly informal methods and easy to use. This paper investigates whether we can replace inspection with testing or not.

Keywords: Software Inspection, Software Quality, Verification, Reliability, Software Cost, Validation, Software Testing.

INTRODUCTION

While Software has become one of the most valuable products of the past decades, its growing complexity and size is responsible for making it one of the most challenging one to build and maintain. The challenge stems from the fact that software development belongs to the most labor and, at the same time, knowledge- intensive processes of today's world. The heavy dependence on knowledgeable human beings may be one reason why software development is often compared to an art or craft rather to an engineering discipline. However, it has almost become impossible nowadays for a craftsman to produce large software system according to a given schedule, to a limited budget, and to the quality requirements of a customer at delivery. Hence researchers as well as practitioners are increasingly obliged to address the question of how to integrate engineering principles into software development. An important one is to perform high quality enhancing activities as early as possible. Despite the simplicity of the principle one can observe in the software industry that the activity of detecting and correcting software problems is often deferred until late in the project.

To address this issue, engineering-oriented [1] software organizations have started to implement rigorous software inspection and software testing.

Software inspection is a proven method for improving software product quality and it provides a very cost effective way to improve their development process. Software inspection allows software development teams to find effect earlier and cheaper, thus reducing rework cost. In addition there are often benefits more difficult to quantify. Software inspections aid in project management; and they provide more definite and more dependable milestones.

Performing an inspection immediately after completion of a work product, or a part, and analyzing the resultant data of the detected defects will provide an early quality indicator to the management and technical team.

Software testing is the process of executing a program or system with the intent of finding errors. In other words, software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Software testing is involved in every stage of software life cycle, but the testing done at each level of software development is different in nature. There are different types of software testing techniques viz. Unit Testing, Integration Testing, System Testing and Acceptance Testing. Software testing has been widely used as a way to help engineers develop high quality systems.

The contribution of this study is first, a view of the software inspection and software testing. Second, the study checks whether can we replace software inspection with software testing?

DEFINITION

Software Testing

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test [6], with respect to the context

[*] MCA Deptt., Guru Nanak Khalsa Institute of Technology & Mangement Studies, Yamuna Nagar, Kurukshetra University, Kurukshetra, Haryana, India. *E-mail: mitu.kajal@gmail.com*

[**] MCA Deptt., Guru Nanak Khalsa Institute of Technology & Mangement Studies, Yamuna Nagar, Kurukshetra University, Kurukshetra, Haryana, India. *E-mail: asarchusharma@gmail.com*

[***] MCA Deptt., Guru Nanak Khalsa Institute of Technology & Mangement Studies, Yamuna Nagar, Kurukshetra University, Kurukshetra, Haryana, India. *E-mail: vipsmax@gmail.com*

in which it is intended to operate. Software testing is used for different purposes such as

- To improve quality
- For verification & validation
- For reliability estimation

Software Inspection

The word 'inspect' is an ordinary English verb whose meaning is "to look at or examine". Inspection in software engineering [2] refers to peer review of any work product by trained individuals who look for defects using a well defined process. An inspection is also known as Fagan Inspection after Michael Fagan, the inventor of the process. Inspections are a static technique in that the code or document is not executed. Each inspected document during the project life cycle is examined and compared to a previous state to see if the transformed state has been correctly transformed and is itself correct. Following Figure 1 is an example of this relationship for inspection :

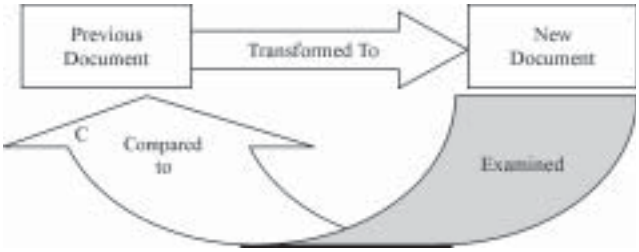

Figure 1: Relationships of Documents for an Inspection Inspection Process

A formal inspection consists of several activities which are as follows:

1. Planning & Scheduling:

The moderator selects the inspection team, obtain material to be inspected from the producer and distribute them and any other relevant documents to the inspection team in advance. Materials should be distributed at least at least 2 or 3 days prior to the inspection. The complete planning and scheduling for inspection occurs in two stages.

a. When the project leader defines the initial project plan(inspection planning).

b. When specific work product approach inspection readiness (inspection scheduling).

2. Overview:

The overview meeting is schedule based on a need as determined by the moderator with the project leader and producer. This includes education and transfer of information necessary for the participant to perform an effective and efficient inspection. If an inspection is being used to provide the education or backup capability then an overview is usually warranted.

3. Preparation:

Each participant is responsible for examining the work product to the actual inspection meeting, noting any defects found or issues to be raised. Almost 75% of the errors found during inspection are identified during the preparation step. The product should be compared against any predecessor (specification) documents to assess completeness and correctness. Checklist of defects commonly found in this type of work product should be used during preparation to hunt to anticipated types of errors.

4. Inspection Meeting:

The inspection meeting is the heart of software inspection. Its primary purpose is to find as many defects as possible during the meeting. During the discussion, all inspectors can report defects or raise other issues, which are documented on a form by the recorder.

The meeting should last no more than two hours. At its conclusion, the group agrees on an assessment of the product: accepted as it is or accepted with minor revisions needed and a second inspection required or rebuild the product.

5. Analysis Meeting:

This analysis meeting step involves understanding. This activity was not included with the original inspection method defined by Fagan. Carole Jones had identified it as adding value and improvement when built upon the traditional inspection method. This activity is viewed as optional by many inspection proponents, but it is highly recommended. If included, there is some added cost for inspections.

6. Rework:

The producer is responsible for resolving all issues raise during the inspection. This does not necessarily mean making every change that was suggested, but an explicit decision must be made about how each issue or defect will be dealt with.

7. Follow-up:

To verify that the necessary rework has been performed properly, the moderator is responsible for following up with the author. If a significant fraction (say 10%) of the work product was modified, an additional inspection may be required. This is the final gate through which the product must pass in order to the inspection to be completed.

8. Prevention Meeting:

The prevention team leader for the prevention meeting will record the results of the meeting & deliver proposals for actions to the organization management The prevention meeting as part of the inspection is considered optional and is dependent on the analysis meeting. This is the final activity in the evolved inspection meeting.

Software Testing

There is a plethora of testing methods and testing techniques, serving multiple purposes in different life cycle phases. Classified by purpose, software testing can be divided into: correctness testing, performance testing, reliability testing and security testing. Classified by life-cycle phase, software testing can be classified into the following categories: requirements phase testing, design phase testing, program phase testing, evaluating test results, installation phase testing, acceptance testing and maintenance testing. By scope, software testing can be categorized as follows: unit testing, component testing, integration testing, and system testing.

1. Correctness Testing

Correctness is the minimum requirement of software, the essential purpose of testing. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or may not know the inside details of the software module under test, *e.g.* control flow, data flow, etc. Therefore, either a white-box point of view or black-box point of view can be taken in testing software. We must note that the black-box and white-box ideas are not limited in correctness testing only.

- **Black-Box Testing**

  The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven [3], or requirements-based [4] testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing—a testing method emphasized on executing the functions and examination of their input and output data.

- **White-Box Testing**

  Contrary to black-box testing, software is viewed as a white-box, or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. Testing plans are made according to the details of the software implementation, such as programming language, logic, and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing [3] or design-based testing [4].

Control-flow testing, loop testing, and data-flow testing [5], all maps the corresponding flow structure of the software into a directed graph. Test cases are carefully selected based on the criterion that all the nodes or paths are covered or traversed at least once. By doing so we may discover unnecessary "dead" code — code that is of no use, or never get executed at all, which can not be discovered by functional testing. In mutation testing, the original program code is perturbed and many mutated programs are created, each contains one fault. Each faulty version of the program is called a mutant.

2. Performance Testing

Performance has always been a great concern and a driving force of computer evolution. The goal of performance testing can be performance bottleneck identification, performance comparison and evaluation, etc. The typical method of doing performance testing is using a benchmark—a program, workload or trace designed to be representative of the typical system usage [7].

3. Reliability Testing

Software reliability refers to the probability of failure-free operation of a system. The robustness of a software component is the degree to which it can function correctly in the presence of exceptional inputs or stressful environmental conditions. Robustness testing [8] differs with correctness testing in the sense that the functional correctness of the software is not of concern. It only watches for robustness problems such as machine crashes, process hangs or abnormal termination.

4. Security Testing

Software quality, reliability and security are tightly coupled. The purpose of security testing of these systems include identifying and removing software flaws that may potentially lead to security violations, and validating the effectiveness of security measures. Simulated security attacks can be performed to find vulnerabilities.

Can Software inspection replace software testing?

Inspections are low technology, labor intensive and rarely fun. All these factors cause people to question the value of inspection. A frequent challenge is put forwarded that some form of testing such as unit testing, will be just as effective & effective as that inspection.

If it is accepted that inspection have value as compared to testing , then a next question will be arise that , should we combine testing & inspection , in order to get better

results. But there are some reasons for not testing before inspections. As inspection require a motivated team, testing first may lead to a view that the code is reasonably stable & the team will be less motivated to perform the best inspection.

1. With the investment of test the producer may have fewer tendencies to receive the major rework on an already stable program that will also require retesting.

There are also so many disadvantages of doing software testing before the software inspection which are as follows:-

1. Unit testing leads the software developer to have false confidence that the product works, so why we should perform software inspection.
2. It is hard decision to inspect a large batch that has been tested and there may the view that there is no longer time to inspect.

There are so many reasons to perform the software inspection before the software testing due to the following reasons:

1. You can bypass the unit test if the software inspection produces very good results. You can recover earlier with lower cost to serious design defects found in software inspection versus software unit testing.

Following are some experimental result that compare the software inspection with software testing.

Table 1
Finding Different Kinds of Bugs by Code Inspection or Testing

Sr. No.	Error Type/Location	Software Inspection	Software Testing
1	Module interface errors	X	-
2	Excessive code complexity	X	-
3	Unrequired functionality present	X	-
4	Usability problems	-	X
5	Performance problems	X	X
6	Badly structured code	X	-
7	Failure to meet requirements	X	X
8	Boundary value errors	X	X

By analyzing the above table, we found that software inspection found 2.7 times more errors as compared to unit test. It is also found in some experiments that a defect caught by testing cost 14.5 times as much as to do one found by software inspection.

Inspection tends to reveal different kinds of errors than do testing activities. Table 1 shows some common types of programming problems and techniques which are effective for detecting errors.

The use of software code inspections, design inspections and requirements inspections has been found to increase software quality and lower software development cost.

The inspection process includes the collection of data with which one can analyze the effectiveness of the process. There are many articles reporting the results of such analysis. Some researchers have summarized these reports. They define defect detection effectiveness as the percentage of defects in an artifacts discovered by the detection techniques. The following table shows the average efforts to detect defects in hours per defect.

Table 2
Average Efforts to Detect Defects in Hours Per Defect

Sr. No.	*Defect Detection Technique*	*Min. Value*	*Most Likely Value*	*Max. Value*
1	Design inspection	0.58	1.58	2.9
2	Code inspection	0.67	1.46	2.7
3	Software testing	4.5	6.0	17

CONCLUSION

Testing and inspection are not mutually exclusive; instead they complement each other as quality assurance techniques, both improving different aspects of product quality. Fagan (1976) reports that inspection pays off even if the same code later goes through testing. Inspection finds different kinds of errors than testing finds the errors.

Finding defects is not the only goal of testing, for example, testing is still needed to assess reliability. Finally we can say that we can't replace software inspection with software testing, but both of these are two faces of a coin.

REFERENCES

[1] Software Engineering, *IEEE Computer Society Press,* (1998) 340-349.

[2] A Standard for Inspection Application Software, William E. Perry, (1990).

[3] Myers, Glenford J., *The Art of Software Testing,* Publication Info: New York : Wiley, c 1979. ISBN: 0471043281 Physical Description: xi, 177 p.: ill.; 24 cm.

[4] Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication Info: Wellesley, Mass. : QED Information Sciences, (1988). ISBN: 0894352423. Physical Description: ix, 280 p. : ill ; 24 cm.

[5] Norman Parrington and Marc Roper, *Understanding Software Testing,* Published by John Willey & Sons, (1989). ISBN: 0-7458-0533-7; 0-470-21462.

[6] *http://en.wikipedia.org/wiki/Software_testing.*

[7] Filippos I. Vokolos, and Elaine J. Weyuker; Proceedings of the First International Workshop on Software and Performance, (1998) 80-87.

[8] IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610. 12-1990), *IEEE Computer* Soc., (1990).

[9] *Software Inspection*, Ronald A. Radice, Tata MacGraw Hill, (2003).

[10] *Software Testing* , Ron Patton, TechMedia.

[11] *Software Testing Techniques*, Boris Beizer, DreamTech Press (2005).