Software Engineering 2012, 2(4): 160-164
DOI: 10.5923/j.se.20120204.09

Software Defect Management Using a Comprehensive Software Inspection Model

Navid Hashemi Taba*, Siew Hock Ow

Department of Software Engineering, University of Malaya, Kuala Lumpur, 50603, Malaysia

Abstract Traditional inspection approaches that are used for more than three decades are not effective for current software and development processes. The studies and experiments by testing and inspection professionals showed that customizing inspections can increase their effectiveness as well as efficiency. The comprehensive software inspection model in this article performs defect removal actions as an important duty of inspection, as well as, using the capabilities of collaborative and knowledge base systems. The process improvement is continuously in progress by creating swap iteration in inspection model kernel. In order to validate the model, it is implemented in a real software inspection project. The varieties of detected and removed defects show the potential performance of the model.

Keywords Software Inspection, Inspection Model, Defect Management, Knowledge Base

1. Introduction

Software inspection is considered by scholars as an engineering and economic approach for software debugging and software qualification improvement. Fagan is known as the founder of software inspection[1, 2]. In 2002 Frank and others issued a paper with inspection subject. They believed that a common and successful technique used for examining traditional specifications is inspection[3]. Although in recent years code inspection by using automated tools has overcome the formal methods of software documents' review[4]. Another facility of software inspection is using collaborative tools in a distributed manner. Using these tools creates a possibility of tele-working for inspectors who are in different time zones and locations. This electronically collaboration is a proper replacement for traditional approaches of gathering the inspectors in one location. The common disadvantage of inspection models is removing identified defects through the inspection mechanisms. This means that the inspection methods advance up to discover the defects and addressing their causes. However, the final goal of inspection is defect removal not just defect detection. The proposed comprehensive model involves defect removal procedures as a major part of inspection process. The implementation of the model on a quality control project shows its capabilities in defect detection and removal.

* Corresponding author:
nhtaba@siswa.um.edu.my (Navid Hashemi Taba)
Published online at http://journal.sapub.org/se

2. Defect Management

Managing the defects is so important for succession the inspection process. Finding defects based on the predefined defect patterns is the common essential task for any inspection model. The first and the most important phase of software inspection is individual preparation. In other words, inspectors must be utterly familiar with development environment, development tools, Projects characteristics, and software product. Regardless of special kind of technique or method to distinguish and remove the defect, the sequence diagram in Figure 1 shows its general strategy[5].

Figure 1. General Strategy for Software Inspection Process

2.1. Defect Definition

According to IEEE, a work product possesses a defect when it faces some shortcomings and inadequacies during providing its own requirements and attributes. Therefore 'repairing', 'reworking' or 'replacing' is necessary for its removal. According to IEEE, a work product possesses a defect when it Standard IEEE dictionary has defined 'defect', 'fault' and 'failure' specifically. These definitions and their elated references are represented in Table 1 Following table has offered different definitions and examples for defect attributes. There are different classifications of defects. Two main classes are Omission and Commission that in the

former omitted elements are considered as a defect and in the latter; those elements, which exist but are incorrect, are considered as defects.

Table 1. Glossary of Terms and Definitions

Term	Definition	Reference
Defect	An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced.	Project Management Institute
Error	A human action that produces an incorrect result.	ISO/IEC 24765:2009
Failure	Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits.	ISO/IEC 25000:2005
	An event in which a system or system component does not perform a required function within specified limits.	ISO/IEC 24765:2009
Problem	Difficulty or uncertainty experienced by one or more persons, resulting from an unsatisfactory encounter with a system in use.	ISO/IEC 24765:2009
	A negative situation to overcome.	ISO/IEC 24765:2009

2.2. Interconnected Relations among Problem, Failure, and Defect

Figure 2 depicts the interconnected relation among the problem, failure and defect failure. The rounded rectangle is used to show the entities and some links connected them to each other. The symbols, which are used at the end of links, represent the number of entities[6].

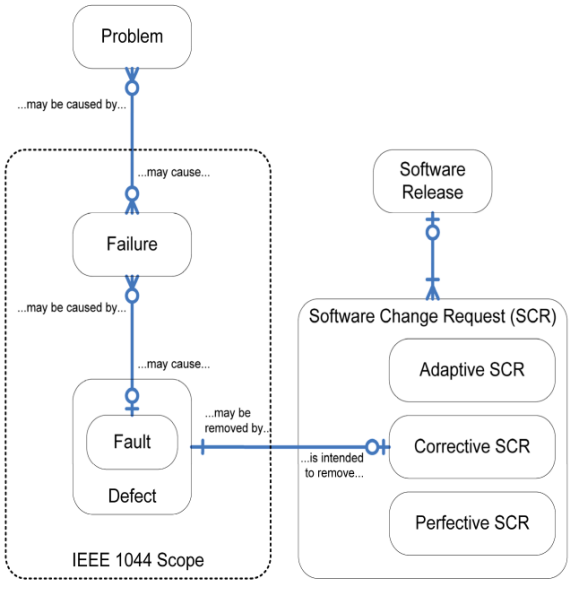

Figure 2. Interconnected Relations

A hollow circle at the end of a link clarifies that the entities are optional, in other words, their existence is not necessary. Trident symbols represent this point that several entities may attend in the connection. Lack of a trident symbol, means that utmost one entity could be applied. Two interconnected rounded rectangles show the relation between the child and parent; the outside symbol represents the parent and the inside one the child. For instance, as it can be seen, defect will be the parent and failure will be the child.

3. Related Works

Tyran stated that software inspections have been found to be one of the most effective ways to promote quality and productivity in software development[7]. The researcher emphasized the correction of a defect found early in development has 10 to 100 times less cost to remove comparing rework performed at the latter stages. According to Suma, Nair, and Gopalakrishnan, the key challenge of an IT industry is to design a software product with minimum post deployment defects[8]. Armour in a test oriented paper, stated that inspection is a way to obtain a high-quality of software[9].

Zheng et al.[10] introduced Automated Static Analysis (ASA) to correct failures before inspections or clients reports or doing some tests lead to their discovery. In his invaluable paper, he has analysed the statistical analysis results of a case study to do with ASA. The researcher has demonstrated that static analytical tools account for as a complement for other error detection techniques and lead to economical development of software product with a high quality.

Leite, Julio in 2005]11] issued an article that shows how inspections help software developers to better manage the production of scenarios. They have used Fagan's inspections as the main paradigm in the design of our proposed process. The process was applied to case studies and data were collected regarding the types of problems as well as the effort to find them. During a case study, software products of Nortel networks with more there 33 million line program was inspected. The objective of the research was that whether ASA can make qualitative improvement of software products in an organization. As an important step, classification of different types of defects and their correspondence with software development process stages are performed and its summary, adopted[12], is available in table 2.

Table 2. ODC Defect Types and Process Associations

Process Association	Defect Type
Design	Function
Low Level Design	Interface, Checking, Timing/Serialization, Algorithm
Code	Checking, assignment
Library Tools	Build, Package, Merge
Publications	Documentation

4. Proposed Inspection Model

The inspection model composed of four important phases as illustrated in Figure 3, which are: preparation, defect plan design, generative inspection procedures, and inspection process evaluation.

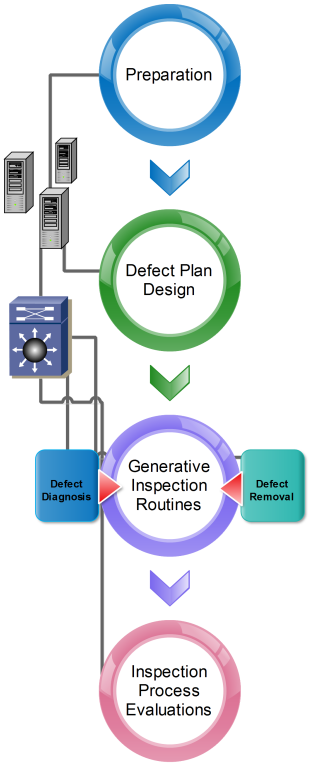

Figure 3. Generative Software Inspection Model

To implement this conceptual model, it is necessary that inspectors, developers, and users communicate during the process using a comprehensive collaboration tool. Also the involved people in inspection process should be electronically trained to be dominant on tools, methods, and inspection artifacts. In order to survey the causes of effects of defects incidence wisely, the model suggests designing and using a knowledge base. The phases of the aforementioned model are explained as below.

4.1. Preparation

The initial point for inspection process is preparing the environment and inspectors. In this phase, the first step is to select expert inspectors according to their required skills and inspected artifacts specifications. The second step is to arrange inspection team including team organization structure in a centralized, decentralized, or distributed and identifying responsibilities, roles, and duties of each member of the team. The third step is to distribute specifications and, in some cases, the artifacts between team members or making them available for inspection team. The last step is to do a quick flash test to be conversant of inspectors' situations and knowledge.

4.2. Designing Defect Plan

Proper resource allocation, scheduling and goals determination are crucial in the success of an inspection process[9]. The suggested steps of this phase, initiate by defining profile and access right for inspectors. However, the next steps are: Defining an appropriate scheduling and a complete charter including collaboration method and resolving possible disagreements, determining milestones ad finally collaboration protocols.

4.3. Generative Inspection Procedures

This phase includes doing the repetitive procedures of two complementary sets:

- The first action set that is called Detect Diagnosis (DD) contains the required actions to identify and recognize the defects. Performing inspection procedures, defect detection, explaining the details of each defect, sketching the cause and effect diagrams, and updating the defects' databases are the main actions of DD set in the third phase.
- Defect Removal (DR) is the second routine set that is considered as the most specific attribute of the proposed model and makes the model intelligent and generative, also satisfies the main goal of inspection process, which is defect removal. The other supplementary duties of this phase are removing defects from artifacts and preparing a new version, updating related documents, defect plan and finally creating defect report.
- DD and DR swapping: as it is mentioned earlier, two sets that form intelligent inspection must be run iteratively and periodically. The iteratively execution feature makes it possible to remove new arisen defects while detecting the other defects. The key factor is to recognize when the cycle should be broken and entered the last phase. However, these should be considered in defect plan as certificate instruction and termination criteria.
- Defect knowledgebase: the action of the two sets, DD and DR, are done by using a knowledgebase composed of defects related rules and facts. In this base the potential defects and causes are stored and by detecting each defect, the inspectors establish, reform or modify the rules. Using an inference engine may help the inspectors to do their duties. The aforementioned inference engine reminds about the possible defects and shows the possible causes (if any defect found).

4.4. Inspection Process Evaluation

As there is a specific plan for each inspection, the evaluation process should be done according to a specific plan so the first step of the last phase is to customize evaluation metrics[13]. The second step is to finalize the evaluation formulas according to pre-defined criteria. Respectively, next step is to put data in the related formula and analysing them. The results of these evaluations can be useful in future inspection plan designing and improving the methods used in evaluation. It adds the learning property to the system that is the special attribute of an intelligent model.

5. Model Implementation

To have a real evaluation of model performance, software quality control project of an auto spare part company is selected. Therefore, some system development documents and artifact related to different processes like Purchase, Sales, Production, and Maintenance were inspected. Table 3 shows the inspected processes and efforts for defect detection and defect removal using proposed model in developing each process.

Kaplan stated that defect detection in early phases of system development dramatically reduces the quality costs[14]. Adapting aforementioned research, the minimum and maximum cost saving rate as the model performance criteria are calculated and presented in table 4.

The minimum saving is related to defect detection in immediate next phase and maximum performance is due to detection the defects in last phase of development or after shipping to customer.

Table 5 shows total saved effort as the model efficiency criteria for defect detection and removal. As it is clear, for some defects the required efforts for defect detection is more than the necessary efforts for detect removal and in some cases this fact is reverse.

Table 3. Defect Detection and Removal Efforts

Quality Management Processes	Efforts for Defects Detection and Removal in Case Study (Person / Week)											
	Analysis			Design			Code			Test		
	IE	DDE	DRE	IE	DDE	DRE	IE	DDE	DRE	IE	DDE	DRE
Document and data Control, Analysis of data, Preventive and Corrective Action	7	4	3	4	1	3	5	2	3	6	3	3
Sales, Customer Related Processes	5	2	3	4	2	2	6	5	1	5	2	3
Purchase	4	2	3	3	1	2	9	7	2	7	3	4
Process Audit	7	4	3	8	3	5	7	6	1	6	4	2
Planning of production	8	4	4	9	5	4	8	4	4	10	5	5
Training	6	4	2	4	2	2	6	4	2	2	1	1
Storage	8	3	5	6	3	3	10	3	7	9	5	4
Product Audit	10	4	6	12	5	7	9	4	5	14	6	8
Inspection and test (Lab, Calibration)	6	4	2	8	6	2	6	4	2	5	3	2
Maintenance & Tool management	8	3	5	6	4	2	8	4	4	6	4	2
Management review, QMS Planning	12	6	6	7	2	5	4	2	2	14	5	9
TOTAL	81	40	42	71	34	37	78	45	33	84	41	43

IE: Inspection Effort, DDE: Defect Detection Effort, DRE: Defect Removal Effort

Table 4. Defect Amplification

	Relative Cost of Correcting Defects in Next Phases									
Phases	**Analysis**		**Design**		**Code**		**Test**		**Implementation**	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
Analysis	Base		3	6	10	10	15	70	40	1000
Design	Base				2	3	4	17	10	250
Code	Base						2	7	4	100
Test	Base								2	20

Table 5. Total Saved Efforts Defect

Phase	IE	DDE	DRE	AAR	EDA	SE
Analysis	81	40	42	144	6000	5200
Design	71	34	37	62	2200	2130
Code	78	45	33	28	924	846
Test	84	41	43	11	473	389
Total	314	160	155	-	9597	8565

IE: Inspection Effort Defect; DDE: Detection Effort; DRE: Defect Removal Effort; AAR: Average Amplification Rate; EDS: Expected Defect Saved

6. Involved People in the Model

Users, software developers, independent and internal inspectors are the involved people in inspection process. Using web-based distributed tools and collaboration framework not only leads to inspection process facilitation, but also removes the gap and overlaps of the actions done. Another advantage of using this kind of environments is involving inspection process employers who are in different time zones and geographically in far positions.

Bryksyzenski[15] stated: *"Software inspection has decisively improved software quality, development cycle time, overall maintainability.*" Finally we should say that the integrated environments supported by relation or networking database are better to present the experiments between the projects, and then do some traditional document based approaches.

7. Conclusions

The proposed model in this research, suggested a defect management approach systematically detect removing through an iterative manner. Registering the events and rules related to defects and their causes in a knowledgebase, makes the model intelligent. Using distributed collaboration tools enables software inspectors to do their duties without any gap and overlap. Customized evaluations of inspection process prepare useful information about performance and effectiveness of inspection process, which causes continuous improvement in the next iterations of a project lifecycle. Implementing the model in a real environment to detect and remove the real defects shows the performance of the model. Developing and maintaining collaboration tools is highly recommended to gain better performance.

REFERENCES

[1] M. E. Fagan, "Design and Code Inspection to Reduce Errors in Program Development," IBM Systems Journal, vol. 15, No. 3, pp. 182-211, 1976.

[2] M. E. Fagan, "Advances in Software Inspections," IEEE Trans. on Software Engineering, SE, vol. 12(7), pp.744-751, July 1986.

[3] H. Frank, S. Thilo, E. Dietmar, "Defect Detection for Executable Specifications — An Experiment, "International Journal of Software Engineering & Knowledge Engineering, , vol. 12, Issue 6, pp. 637, Dec2002.

[4] M. Bertrand, "Design and Code Reviews in the Age of the Internet, "Communications of the ACM, vol. 51, Issue 9, pp. 66-71, Sep 2008.

[5] ISO/IEC 24765-2009, Systems and Software Engineering—Vocabulary.

[6] IEEE Std 1044™-2009(Revision ofIEEE Std 1044-1993).

[7] C. K. Tyran, "A Software Inspection Exercise for the Systems Analysis and Design Course,"Journal of Information Systems Education, Vol. 17, Issue 3, pp. 341-351, Fall2006.

[8] V. Suma, T. Nair, R. Gopalakrishnan, "Effective Defect Prevention Approach in Software Process for Achieving Better QualityLevels,"Proceedings of World Academy of Science: Engineering & Technology, Vol. 32, pp. 288-292, Aug 2008.

[9] P. G. Armour, "The Unconscious Art of Software Testing, "Communications of the ACM, vol. 48, Issue 1, pp. 15-18, Jan2005.

[10] Zheng, J.; Williams, L.; Nagappan, N.; Snipes, W.; Hudepohl, J.P.; Vouk, M.A., "On the value of static analysis for fault detection in software," Software Engineering, IEEE Transactions on , vol.32, no.4, pp. 240- 253, April 2006doi: 10.1109/TSE.2006.38

[11] Leite, Julio; Doorn, Jorge; Hadad, Graciela; Kaplan, Gladys, 2005. Scenario inspections. Leite, Julio; Doorn, Jorge; Hadad, Graciela; Kaplan, Gladys. Requirements Engineering, Feb2005, Vol. 10 Issue 1, p1-21.

[12] R. Chillarege, I.S. Bhandari, J. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M.Y. Wong, "Orthogonal Defect Classification—A Concept for In-Process Measurements," IEEE Trans. Software Eng., vol. 18, no. 11, pp. 943-956, Nov. 1992.