International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:12, 2015

Defect Management Life Cycle Process for Software Quality Improvement

Aedah A. Rahman, Nurdatillah Hasim

***Abstract*—**Software quality issues require special attention especially in view of the demands of quality software product to meet customer satisfaction. Software development projects in most organisations need proper defect management process in order to produce high quality software product and reduce the number of defects. The research question of this study is how to produce high quality software and reducing the number of defects. Therefore, the objective of this paper is to provide a framework for managing software defects by following defined life cycle processes. The methodology starts by reviewing defects, defect models, best practices, and standards. A framework for defect management life cycle is proposed. The major contribution of this study is to define a defect management roadmap in software development. The adoption of an effective defect management process helps to achieve the ultimate goal of producing high quality software products and contributes towards continuous software process improvement.

***Keywords*—**Defects, defect management, life cycle process, software quality.

I. Introduction

MOST software organisations or IT departments are involved in the use of defect management process to improve the efficiency of the software development projects. Software quality is an important aspect in software development which ensures quality software product is developed. According to [1], software quality is defined as the degree of conformance to specific functional requirements, specified quality standards, and good software engineering practices. The IEEE definition of software quality [2], are stated as (1) the degree to which a system, component, or process meets specified requirements; (2) the degree to which a system, component, or process meets customer or user needs or expectations. However, in dealing with software engineering activities, software quality problems or bugs may occur along the duration that the product is being developed. The types of problems may be referred as software errors, faults, or failures. A small defect may cause damage in terms of monetary, reputation and loss of customer trust. The defects may reduce the software product quality and customer satisfaction. Defect which is destructive in nature is deficiency in the software product will surely reduce the software product quality. Hence, affect the efficiency and quality of software engineering processes. Most importantly, the defect in an application may give negative impact to all phases of software development process such as in software requirements engineering, software design, software implementation, software testing and software maintenance phases. Table I described on software error, fault, and failure [2]. The IEEE definitions error, fault, and failure are also given in Table I [3].

A. A. Rahman and N. Hasim are with the Software Engineering Section, Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Jalan Sultan Ismail, 50250 Kuala Lumpur, Malaysia (e-mail: aedah@ unikl.edu.my, nurdatillah@ unikl.edu.my).

II. Related Research Works

This section describes the related research works approach that is relevant to this study. Table I states the definition of each software quality problem.

TABLE I
Software Quality Problem

Software Quality Problem Types	Description
Software error	Section of code that are partially or totally incorrect as a result of a grammatical, logical or other mistake made by a systems analyst, a programmer, or member of software development team IEEE definition: Human action that leads to incorrect result
Software fault	Software errors that cause the incorrect functioning of the software during a specific application IEEE definition: Incorrect decision taken when understanding the given information
Software failure	Software faults become software failures only when they are activated, when a user tries to apply the specific software section that is faulty. The root of any software failure is software error. IEEE definition: Inability of a function to meet the expected requirements

Galin [2] defines nine causes of software errors which are faulty requirements definition, client-developer communication failures, deliberate deviations from software requirements, logical design errors, coding errors, non-compliance with documentation and coding instructions, shortcomings of the testing process, procedure errors, and documentation errors.

This paper investigates on the management of these defects by defining the process life cycle. The review is performed on defect management tool, defect process, standards, and best practices. Rahman [4] investigated on the needs of a framework for defect management system. In which the defect tracking process is implemented by using a web-enabled defect tracking system that allow project management, development, quality assurance and management of software problems. This indicates that defect tracking is an important practice that should be considered for the proposed model. Hasim and Rahman [5] identified the metric used for defect management that is the defect density. The calculation used is based on the measure of the number of total defect found

divided by the size of the system measured. The system size is measured by lines of code (LOC). This indicates that there is a need for defect prediction and prevention processes.

The review on existing standards and best practices are also conducted. The Capability Maturity Model (CMM) defined five maturity levels of and their key process areas (KPAs) [6], [7]. The maturity levels are:

1) Level 1: Initial
2) Level 2: Repeatable
3) Level 3: Defined
4) Level 4: Managed
5) Level 5: Optimizing**

Level 5 contains the KPA which is related to defect management, the Defect Prevention (indicated by the symbol **). Capability Maturity Model Integration (CMMI) which is a newer process improvement framework also includes a relevant process areas (PAs), known as Causal Analysis and Resolution (CAR) [8]. CAR is indicated at level 5 (Optimizing) (refer to the symbol **). The five maturity levels of CMMI are:

1) Level 1: Initial
2) Level 2: Repeatable
3) Level 3: Defined
4) Level 4: Quantitatively Managed
5) Level 5: Optimizing**

ISO/IEC 20000 is an information technology service management standard [9], [10]. The review on ISO/IEC 20000 indicates the existence of processes such as Incident Management and Problem Management which are related to defects model.

III. Research Approach

This section describes the research approach that is being used to conduct this study. This study begins with literature review by finding the definition of defects, and any existing defect models. We have discussed this stage under Related Research Work as stated in the section above. The discussion includes defects, defect models, best practices, and standards. The second stage involves the analysis of requirements and architectures of the defect management process. The third stage is the implementation of defect management life cycle. The research approach is described in Fig. 1.

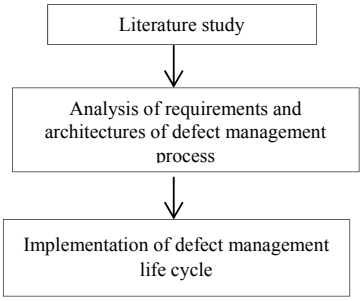

Fig. 1 Research approach

IV. Analysis of Requirements and Architectures of Defect Management Process

The requirements and design for defect management process includes comparing the existing standards and best practices with the proposed framework. This stage determines what should be included in the framework and the design of the framework. The CMM, CMMI, and ISO/IEC 20000 have been reviewed. The review indicates that the defect related processes are included in a higher maturity level of process improvement framework. However, in the case of ISO/IEC 20000, there is no specific cycle has been introduced for this framework. Therefore, there is a need to derive a better life cycle based approach for defect management process.

Table II indicates further key process areas/process areas or processes based on three standards or best practices that are relevant for the implementation of the defect management life cycle. The first column indicates the standards or best practices. Column 2 shows the incident, defect or problem related KPAs/PAs/processes that appears at higher maturity level (ML5). Finally, column 3 states the change, configuration or release related KPAs/PAs/processes that appears at a mixture of lower and higher maturity level (ML2 and ML5).

The final stage of this research, which is the execution of the defect management life cycle, is discussed in the results and findings discussion section. Six steps are involved in the life cycle that will give positive impact to the software quality improvement.

V. Results and Findings: Implementation of Defect Management Life Cycle

Based on the analysis of requirements and architecture conducted in the earlier research phase, the important defect management practices are defect identification, defect analysis, defect prevention, defect resolution, defect monitoring, and defect process improvement. Defect Identification is related to Incident Management in ISO/IEC 20000. Defect Analysis is relevant to Causal Analysis and Resolution (CMMI) and Problem Management (ISO/IEC 20000). The third practice, Defect Prevention can be compared to Defect Prevention (CMM). The change/ configuration/release related KPAs/PAs/processes can be grouped together. The fourth practice, Defect Resolution is associated with Process Change Management and Technology Change Management KPAs in CMM; Configuration Management PA in CMMI; and Change Management, Configuration and Release Management processes in ISO/IEC 20000. This study determined that defect prediction is also important and it can be assigned under Defect Process Improvement, step 6. This is related to CMMI maturity level 5 in Organizational Process Performance (OPP) and Quantitative Project Management (QPM) KPAs.

The important defect management practices are described as follows.

A. Defect Identification

When a defect repeatedly occurs, they need to be identified

and recorded in the system. Defects are identified at two location, at that time detect was initially detected and at the time the detects have been fixed.

TABLE II
COMPARISON ANALYSIS OF BEST PRACTICES AND STANDARDS

Standards and Best Practices	KPAs/PAs/ Processes	
CMM	Defect Prevention (ML5)	Process Change Mgt (ML5) Technology Change Mgt (ML5)
CMMI	Causal Analysis Resolution (ML5)	Configuration Mgt (ML2)
ISO/IEC 20000	Incident Mgt Problem Mgt	Change Mgt Configuration Mgt Release Mgt

B. Defect Analysis

Defect analysis includes defect classification and taxonomy by using specific naming schemes. The classification of defects represents information such as phases and activities of the software development phases that the defect injection occurs. The framework to evaluate different defect taxonomies is produced in [11]. A meta taxonomy comparing the different types of taxonomies are developed and categorized into three aspects attributes, structure type and properties [11]. The taxonomy contains attributes as stated below.

1) Location
2) Timing
3) Symptom
4) End result
5) Mechanism
6) Cause (error)
7) Severity
8) Cost it

C. Defect Prevention

In this phase, the defects are investigated. Diagnosis is performed on the underlying causes of defect. Root cause analysis is conducted at this phase and preventive measures are taken to avoid the detects from happening again.

D. Defect Resolution

This process involves request for change and resolving the defect.

E. Defect Monitoring

Defect monitoring involves ensuring the defect management process is effectively performed at project level. This step also involves the verification of fixes that have been made to resolve the defect.

F. Defect Process Improvement

Defect process improvement involves the activity to predict potential software defects from test data. The detect prediction contributes to the early detection of defects in software development life cycle. This phase involves software metrics such as the use of defect removal efficiency and defect density [5]. Defect removal efficiency is computed as the number of defects before release of product to the total number of underlying defects.

Fig. 2 illustrates the defect management life cycle which contains the six steps: defect identification, defect analysis, defect prevention, defect resolution, defect monitoring and defect process improvement.

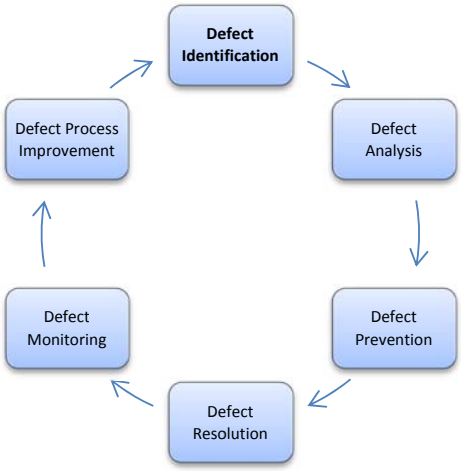

Fig. 2 Research approach

VI. CONCLUSION

This paper has presented a life cycle process for managing defects in software development projects that can be used by organizations. The model can be used in accordance to existing defect management tool available in the market nowadays. In future we would like to look into each phases in defect management thoroughly and how it contributes to software process and quality improvement. In addition, we would like to further investigate and define process metrics especially in the defect prevention and process improvement phases.

ACKNOWLEDGMENT

The authors would like to thanks Universiti Kuala Lumpur for the research support.

REFERENCES

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach* (7th ed.). New York: McGraw-Hill International, 2009.
[2] D. Galin, *Software Quality Assurance: From Theory to Implementation*. England: Pearson Education Limited, 2004.
[3] A. Gupta, J. Y. Li, R. Conradi, H. Ronneberg, and E. Landre, "A case study comparing defect profiles of a reused framework and of applications reusing it", Springer, 2008.
[4] A. A. Rahman, "The framework of a web-enabled defect tracking system", in Proc. IEEE International Conf. on Advanced Communication Technology, Korea, 2004,Volume 2, pp. 609-695.
[5] N. Hasim and A. A. Rahman, "Defect density: A review on the calculation size program", in Proc. 4th International Conf. on Machine Vision: Computer Vision and Image Analysis, Pattern Recognition and Basic Technologies, 2011, Singapore, Proc. SPIE Volume 8350.
[6] M. C. Paulk, B. Curtis, M. B. Chrissis, *The Capability Maturity Model: Guidlines for Improving the Software Process,* MA, Reading: Addison Wesley, 1995.
[7] M. C. Paulk, "How ISO 9001 compares with the CMM", in IEEE Software, vol. 12(1), pp. 74–83.

[8] SEI, "CMMI for Development, Version 1.3, CMMI-DEV, V1.3", No. CMU/SEI-2010-TR-033, Pittsburgh, PA: Software Engineering Institute.
[9] ISO, "ISO/IEC 20000-1: 2005, Information Technology - Service Management - Part 1: Specification", London, UK, 2005a.
[10] ISO, "ISO/IEC 20000-2: 2005, Information Technology - Service Management - Part 2: Code of Practice", London, UK, 2005b.
[11] D. Vallespir, F. Grazioli, and J. Herbert, "A framework to evaluate defect taxonomies", in Proc. XV Argentina Congress on Computer Science, 2009.

Aedah A. Rahman Aedah A. Rahman was born in Kuala Lumpur, Malaysia. A. A. Rahman earned her Bachelor of Computer Science in 1998 and Masters of Computer Science in specialization in software engineering in 2002 degrees from the University of Malaya, Kuala Lumpur, Malaysia. Aedah A. Rahman then obtained her Doctor of Philosophy (Ph.D) in Computer Science from Universiti Teknologi Malaysia, Johor, Malaysia.

She is currently a Senior Lecturer with the Software Engineering Section, Malaysian Institute of Information Technology (MIIT), Universiti Kuala Lumpur, Malaysia. Her job is related to teaching and learning, research, coordination, and administration in university. She is the Program Coordinator for software engineering programs in the university. She is a certified tester and certified requirements engineer. In 1999, she has served Universiti Tun Abdul Razak as Lecturer and Head of Software Engineering Department. She continued her career in academic in Open University Malaysia until 2005. She has worked in various university environments. Her 16 years' experience in academic pursuit, also involves research and academic publication national and international level. Her research interests and area of expertise include requirement engineering and management, software process quality and improvement, software quality and process evolution, real-time and embedded system and software engineering.

Dr. Rahman is a member of IEEE and Malaysian Software Engineering Interest Group (mySEIG).

N. Hasim, is a Lecturer in Software Engineering Section, Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Malaysia.