Establishing a Defect Management Process Model for Software Quality Improvement

Hafiz Ansar Khan

Abstract—**Defect remains in the whole life of software because software is developed by humans and 'to err is human'. The main goal of well-organized software defect management process is to produce quality software with the least number of defects to reduce the impact of problems in the organization. Defect management process includes three levels which are defect detection, defect analysis and defect prevention to eliminate and mitigate the potential defects. At first level test the software work product until the entire defects are identified and fixed. The second level is defect analysis in this level previously identified defects are analyzed and time is spent to look their root causes and why they were not detected earlier. The ultimate goal of third level defect prevention is to prevent the defects from recurring in the future. The research question of this study is how to produce quality software with the least number of defects? A well establish a defect management process is one of the success factors of producing a software system within the time and budget. In this paper author have proposed a defect management process model and finds observations by applying the proposed model in one of the case organization. The major contribution of this study is to establish a defect management process model in an organization to reduce the number of defects and produce a quality software product.**

Index Terms—**Defect management process model, ITIL defect management model, defect detection, defect analysis, defect prevention, radial analysis, ODC, MR classification, ODC-CC, RCA, FMEA, FTA.**

I. Introduction

Defect is destructive in all stages of software development. A defect is a flaw, deficiency or inaccuracy in the software product [1]. Defect remains in the whole life of software; each defect which occurs in the software stages is the defect in that software. Everything associated with defect is a repeated process not a condition or situation. The IEEE defines Error, Fault and Failure as Error: an individual action that guide to inaccurate result. Fault: wrong or incorrect action taking to solve the problem. Failure: inability of a software work product function to meet the anticipated requirements [1].

Establishing a defect management process is an attractive way to improve the software quality. Early detection of defect provides cost and time saving for software projects because developers need to produce less new product versions and bug fixes. Moreover, reduce the number of defects in an application increases the level of customer satisfaction, and reliable software is easy to sell to new customers.

To manage software defects three levels are used defect detection, defect analysis and defect prevention. At first level test software work product until the entire defects are identified and fixed. Although it is not possible to test the software hundred percent means completely. However at this level it is assumed to detect many defects as possible this can be done through static analysis and automated testing tools. The second level is defect analysis in this level previously identified defects are analyzed and time is spent to look their root causes and why they were not detected earlier. The Third level is defect prevention in this process specific techniques for example orthogonal defect classification (ODC) [2], [3] are used to identify the defects and their root causes. The ultimate goal of this defect prevention technique is to prevent the defects from recurring in the future. The defect found in the first two levels can also be used in defect prevention to eliminate the root causes of defects [4].

In this paper the research question is how to produce quality software with the least number of defects to reduce the impact of problems in the organization? Previously most of the research on defect management was paid attention to software companies excluding the customer [5]. In this paper the propose model also observes the defect management from the customer's viewpoint.

The paper is structured as Section II describes related work. Section III shows the propose model. Section IV illustrates the simulation of the proposed model. Section V describes the comparison of the proposed model with previous work. And finally section 6concludes the study.

II. Procedure for Paper Submission

Any defect in the software represents a weakness in the process. Unimportant defects are also no different than critical defects. It's a developer best fortune that prevents a defect to cause a major failure [1]. Even the minor defects in the software create an opportunity for the developer to learn and improve the process by preventing potential major failures. Defect remains in the whole life of software the defect itself might not be a big problem; but the reality that there was a defect is a big problem [1].

Identifying defects in the early phases of software development leads to preventing defects in the later phases of software [6]. The cost and time of a defect found in the later stages of software development process can be very high [6]. Fix cost of later found defects gets higher as the software development progress because of rework done in design, development and testing stages. A software defect that has

Manuscript received December 5, 2012; revised February 13, 2013.

Hafiz Ansar Khan is with Shaheed Zulfikar Ali Bhutto Institute of Science and Technology Islamabad, Pakistan (e-mail: ansar_1182@yahoo.com).

cost 1x to resolve in design phase might cost 100 x to resolve after the software work product is released [7].

In Different studies different author has explored the defect management process. Software Engineering Institute, Information Technology Infrastructure Library and Quality Assurance Institute describe different types of defect management models [5]. Quality Assurance Institute describes that defect management includes six essential elements which are discover a defect, defect prevention, base lining the defects, resolution of defect, defect management process improvement, and problem reporting [5]. The IBM defect prevention model has focused on defect prevention techniques which are defect analysis and defect root cause analysis etc.

A. Defect Prevention

Defects are introduced in the software system somewhere in requirement, design and development phase. If a software defects can be indented out of the software system this will never need time and money to find and fix [5]. Microsoft Encarta 2007 defines the prevention as an activity which stops someone to do something or stop something to do an action [8]. Many defect prevention techniques for example FMEA and FTA are used to prevent defects. FMEA is a technique which concentrates on possible failure modes does not extremely focus on the potential failure root causes events, but the second prevention technique FTA first found potential failure modes and then deeply go into all potential root causes [8]. It will be more effective to perform the FMEA and FTA at the start of the project this will reduce the time and cost spend to fix the defects in the later stages. In CMM at level5 defect management is considered as a key process area to plan defect prevention activities [9].

B. Defect Analysis

The main goal of defect analysis techniques is to analyze defects, identify their root causes and then developing the ways to reduce these defects. Defects are analyzed by using the knowledge learns from the previously discovered defects. Examples of defect analysis techniques are radial analysis [10], orthogonal defect classification (ODC) [2-3], orthogonal defect classification computational code (ODC-CC) [2-3], Modification Request (MR) Classification [11], and root cause analysis (RCA) [8].

C. Limitations of Previous Work

Now different organizations are using defect management model presented by the IT Infrastructure Library (ITIL). But, ITIL model does not describe how to carry out testing and defect management activities in IT service management [5]. Secondly this framework does not consider the customer as a relevant participant of the defect management process.

D. ITIL Model Challenges

Jantti Marko and Miettinen Aki in [5] describe different challenges to ITIL based processes. The problem management processes challenges described in [5] are:

The number of known error idea is not detectable in the existing problem management process

1) No baseline available for known defects
2) It is not easy to combine the ITIL concepts with already existing organization defect management system
3) Defect recorded have many data fields which are hardly ever used
4) The association between testing support is ambiguous (reported defects should have a link to test cases)
5) It is not easy to close many defects with one release because customized versions of product used by many customers
6) In ITIL it is difficult to define right frequency of delivering the defect fixes to different customers

E. Defect Classification Challenges

Various defect classification schemes for example ODC, MR classification and ODC-CC have been proposed previously but none of them become a practice [12]. It is found in different studies that defect classification schemes are difficult to use as a general in practice [13, 14, 15]. They can be used in a specific environment or domain. Stefan Wagner in [12] proposes a set of challenges to defect classification schemes which are

1) Interconnection of defects with different software artifacts The existing classification schemes are available in different dimensions but it is not clear what are the necessary dimensions
2) On what factors defect type distribution depends. Do we have domain specific defect type distribution?
3) The work done on defect classification schemes partly related to software quality models

ODC significant effects on the economics of root cause analysis by reducing the time and it also cover larger defect space particularly when the defect volume is large and the skills of the engineering team are limited [16]. ODC Improve software quality by using readily available data to decrease defects injected and increase defects detected.

The main limitations found in ODC are:

1) ODC cannot classify few defects such as GUI-type and data-type [1]
2) Normally ODC is useful in an organization which has a strong measurement system [4]
3) ODC require the capability to constantly group and analyze data over time; numbers of organizations are at lower maturity levels and they don't have this capability [2]
4) Updating of defect types and associated defect triggers makes it complicated to keep track of source defect data over a long period of time [2]

III. Propose Model

In Different studies different author has explored the defect management process. Software Engineering Institute, Information Technology Infrastructure Library and Quality Assurance Institute describe different types of defect management models [5]. In this study authors have tried to propose a defect management process model. The propose model consist of three subparts which are a collection of defect data, defect analysis and prevention and last part is defect resolution and continuous improvement. Each component of the propose model is described below

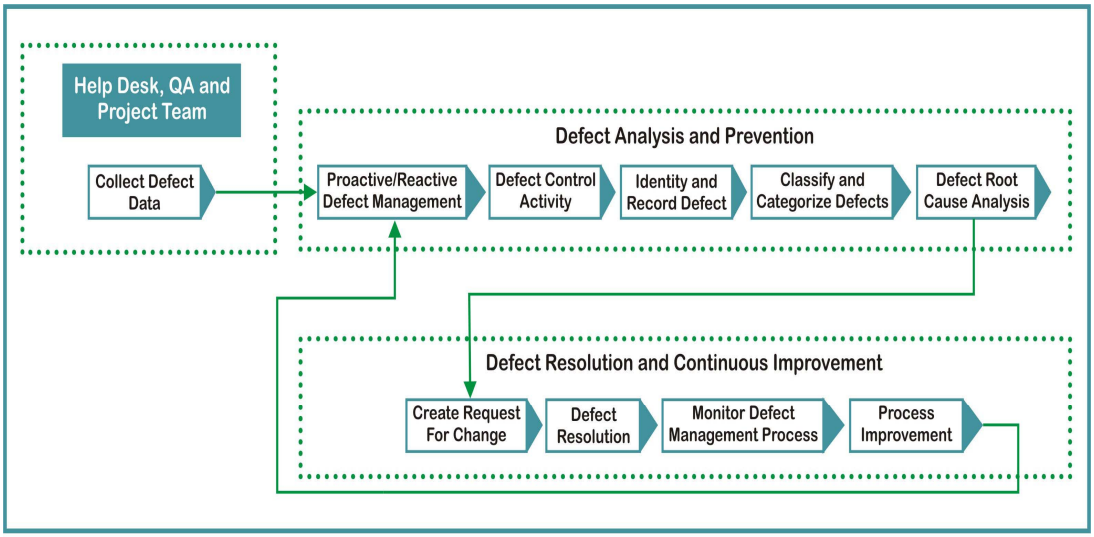


Fig. 1. Software defect management model

A. Help Desk, QA and Project Team

IT organization should collect the defect data from help desk, QA and project team members to achieve a good first time defect fix rate. The goal to collect the defect data from help desk, QA and project teams is to resolve many defects as early as possible before delivering the product to the actual customer.

B. Proactive/Reactive Defect management

No doubt the most excellent approach to defects is to eliminate them as they found. This can be only possible if a defect prevention techniques and processes are used by the organization. The goal of defect prevention is to eliminate the defects altogether so that it cannot re-occur in the future. The primary objective of proactive defect management is to discover and resolve known defects as early as possible before the occurrence of any major problem related to them. Once the defects discovered try to eliminate every defect. For defects that cannot be eliminated try to reduce its impact. The reactive defect management process focuses on identifying the causes of underlying reported problem.

C. Defect Control Activity

Defect control activity starts when the analysis of defect data discloses repetitive problems or the analyzed defect does not match with any of the appearing problem or incident.

D. Identify and Record Defect

When a defect reveals repetitive problems then identify all the defects and record them in the defect management system. The defect record needs to be linked with relevant incident or problem. This will help to identify the defect solution or the work around in the future. A defect has no meaning until the found defect reported and also the developer should acknowledge that the defect is found valid.

E. Classify and Categorize Defect

Different classification schemes are used to classify the defects for example ODC, ODC-CC, MR classification and RCA. Defects are classified by category, priority, urgency and impact. The possible software defect categories for example can be functional, interface and algorithm etc. The impact of the defect is its effect on the organization business and the priority is based on urgency and impact of the defect.

F. Defect Root Cause Analysis

After classification and categorization of defects investigate and diagnose the underlying causes of defect. Different defect analysis techniques, methodology and standard processes are used for root cause analysis.

G. Create Request for Change

Create a request for change to the development team to implement permanent solution for the identified defect.

H. Defect Resolution

Once the developer acknowledges that the found defect is valid then the resolution process starts. While resolving the defect the developer should keep in mind the importance of fixing a defect. After resolution of defects developer must notify to all related parties about the defect status.

I. Monitor Defect Management Process

Project management should continuously monitor the defect management process. Project management should aware the progress of the defect resolution process and impact of defects on customers. The monitoring should be done based on actual requirements defined in the software requirement specification document.

J. Process Improvement

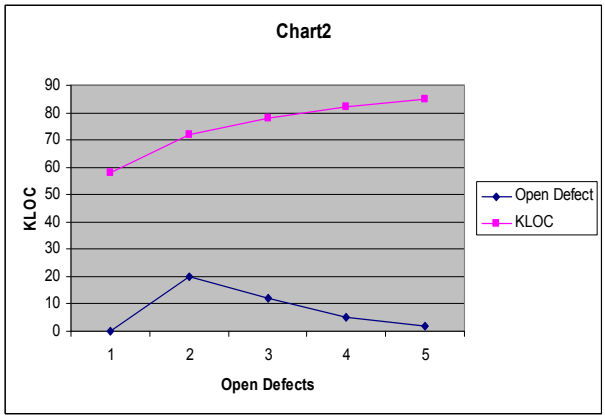
Most of the organizations ignored this process, although this process is one of the big parts of payback. The participants should go back to the phase from where the defect originated and brainstorm what caused the defect. After that they have to review the validation process in which the defect should be caught earlier. This step will not only improve the review process but in fact it will also strengthen the participant's capabilities towards organization business logic.

IV. Simulation of Proposed Model

The authors have applied the propose model in case organization name as 'Moftak Solutions' and simulate the

result of model in the form of chart1 and chart2. Below chart1 shows the relationship between product builds number and defect density where x –axis is the product builds number and y-axis is the defect density. In chart1 first build is the 'baseline' which shows the statistics of the previous round of analysis. Baseline is helpful to begin the defect analysis between different software releases. In 'baseline' product defect density was 6.0 but after applying the propose model in build 1, 2, 3 and 4 it is noticed that the defect density is reduced to 4.3, 3.5, 3.2 and 3.1 in each build respectively.

Chart 2 shows the relationship of 'Open Defects' and 'Kilo Line of Code (KLOC)' used in each build. In the baseline there were 58 kilo lines of code, 72 KLOC in build1, 78 KLOC in build2, 82 KLOC in build3 and 85 KLOC in build4. In the first build there were 20 open defects and in build2 they reduced to 12, in build3 they further reduced to 5 and in the last build4 there were only 2 open defects. This shows the performance improvement.



A. Result

Table I represents the result of four builds. The first row of table 'baseline' shows the statistics of previous round of analysis. The build number is a unique number which identifies the software build. In each build of software 'Open Defects' shows the number of defects which are reported in that build and 'Fix Defect' shows that how many defects are fixed in that build. Known defects in a software builds are identified by using the following formula

$$\text{Known Defects} = KD_{pre} + DR_{cur} - DF_{cur}$$

where 'KD' represents the number of known defects in a previous build, 'DR' stand for number of defects reported in a build and 'DF' represents the number of defects fixed in a build. Defect density can be easily calculated once the numbers of known defects are found in a build. The defect density is calculated by using the following formula

$$\text{Defect Density} = \text{Number of Known defects}/KLOC$$

where 'KLOC' stands for thousands of lines of code.

V. Comparison of Work

Different organizations are using defect management model presented by the IT Infrastructure Library (ITIL) [5]. The major weakness of ITIL model is that it does not consider the customer as a relevant participant of the defect management process. Secondly it also does not specify how to carry out testing and defect management activities in IT service management [5]. It is not easy to combine the ITIL concepts with already existing organization defect management system [5]. The big challenge of ITIL model is lack of performance metrics and knowledge [5].

As comparison to the ITIL model the propose model is easy to use in an organization and secondly it strengthen the organization defect management and review process. In ITIL model customer didn't consider as a relevant participant of the defect management process but in the propose model customer is considered as an active part of the defect management process. Most of the organizations ignored the improvement process which is considered as a process in the describe model because this process is one of the big parts of payback.

TABLE I: Represents The Result of Four Builds

Build	Open Defect	Fix Defect	Known Defect	KLOC	Defect Density
Baseline	-----	-----	350	58	6.0
1	20	58	312	72	4.3
2	12	24	274	78	3.5
3	5	12	267	82	3.2
4	2	7	262	85	3.1
		Total=39	Total=101		

VI. Conclusion

A well establish defect management process is one of the success factor of producing a quality software system. To manage software defects three levels are used defect detection, defect analysis and defect prevention. Currently most of the organizations are using ITIL defect management process model but the major challenges of ITIL model are lack of performance metrics and less participation of customer in the defect management process. In this study authors have tried to propose a defect management process model. The authors have applied the proposed defect management process model in one of the case organization and found that the propose model is easy to use and secondly it strengthens the organization defect management and review process. The major contribution of this study is to establish a defect management process model in an organization to reduce the number of defects and produce a quality software product.

References

[1] A. Gupta, J. Y. Li, R. Conradi, H. Rønneberg, and E. Landre, "A case study comparing defect profiles of a reused framework and of applications reusing it," August 20, 2008, Springer.

[2] A. A. Shenvi, "Defect prevention with orthogonal defect classification," ISEC'09, February 23-26, ACM, India, 2009.

[3] B. Robinson, P. Francis, and F. Ekdahl "A defect-driven process for software quality improvement," USA: New York, ACM, 2008.

[4] A. Andrews, P. Runeson, C. Andersson, T. Thelin, and T. Berling, "What do we know about defect detection methods?" *IEEE Software*, May/June 2006.

[5] J. Marko and M. Aki, "Implementing a software problem management model, a case study," 2006, Springer Link.
[6] N. Agrawal and P. Jalote, "Using defect analysis feedback for improving quality and productivity in iterative software development," March 27, ACM, 2006.
[7] R. Basili and B. Boehm, *Software defect reduction top 10 list*, Los Alamitos, CA, USA: IEEE Computer Society Press, January 2001.
[8] M. McDonald, R. Musson, R. Smith, D. Bean, D. Catlett, L. A. Kilty, and J. Williams, *The practical guide to defect prevention*, Washington: Redmond, Microsoft Press, ch. 11, 2008.
[9] C. P. Chang and C. P. Chu, "Improvement of causal analysis using multivariate statistical process control," January 23, 2008, Springer.
[10] C. Henderson, "Managing software defects: Defect analysis and traceability," vol. 33, July, 2008,.
[11] M. Leszak, D. E. Perry, and D. Stoll, "A case study in root cause defect analysis," June, ACM, 2000.
[12] S. Wagner, "Defect classification and defect types revisited," July 20, 2008, Seattle, Washington, USA.
[13] J. Duraes and H. Madeira, "Defination of software fault emulation operators a field data study," in *Proc. 2003 International Conference on Dependable Systems and Networks*, 2003, IEEE Computer Society.
[14] B. Freimut, C. Denger, and M. Ketterer "An industrial case study of implementing and validating defect classification for process improvement and quality management" in *Proc. 11th IEEE International Software Metrics Symposium (METRICS '05)*, IEEE Computer Society, 2005.
[15] S. Wagner, J. Jurjens, C. Koller, and P. Trischberger, "Comparing bug finding tools with reviews and tests," in *Proc. 17th International Conference on Testing of Communicating Systems (TestCom'05)*, 2005, vol. 3502, Springer.
[16] D. Kelly and T. Shepard, "A case study in the use of defect classification in inspections," in *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, Toronto, Ontario, Canada, ACM, 2001.

Mr. Hafiz Ansar Khan was born at Khushab, Punjab, Pakistan, in 1983. The author has passed MS in software engineering from Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad, Pakistan in 2011. He got his graduation degree BS in software engineering from University of Engineering and Technology Taxilia, Pakistan in 2007. Currently author is working in OA Systems, Islamabad, Pakistan as a Software Quality Assurance Engineer. Previously author had work three years in Moftak Solutions as a Quality Assurance Engineer. The author has total five years' experience in Software Quality Assurance field.