Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Proposed Defect Tracking System for Classifying the Causes of Software Errors

[1]D. Renu, [2]Er. Ritika
[1]RPIIT Technical Campus, Karnal, Haryana India
[2]Asst. Professor CSE Deptt, RPIIT Technical Campus, Karnal, Haryana, India

Abstract: ***This research is concerned with detecting defects in software requirements specification. Motivated by both the problem of producing reliable requirements and the limitations of existing taxonomies to provide a satisfactory level of information about defects in the requirements phase, we focus on providing a better tool for requirements analysts. Only few attempts have been made to classify defects and defect detection techniques. Scattered knowledge about defects and defect detection techniques needs compilation and re-evaluation in order to enhance the ability to discover defects in the requirements phase. Toward this end, this work presents taxonomy of requirements defects and the causes of their occurrences. The purpose is to reach a comprehensive understanding of both the sources of the problem and the solutions of possible defects and defect detection techniques. The taxonomy's design is based on the analysis of each defect and its sources. In addition, this paper proposes a combined-reading technique for defects in requirements. The proposed technique avoids the shortcomings of other reading techniques. The result of applying the recommendations of this work specifically improves the quality of the requirements specification and generally software quality.***

Key words: bugs, defects, bug tracking systems, Errors, Classification.

I. INTRODUCTION

A. Classification of the Causes of Software Errors:

As software errors are the cause of poor software quality, it is important to investigate the causes of these errors in order to prevent them. A software error can be "code error", a "procedure error", a "documentation error", or a "software data error". It should be emphasized that the causes of all these errors are human, made by systems analysts, programmers, software testers, documentation experts, managers and sometimes clients and their representatives. Even in rare cases where software errors may be caused by the development environment (interpreters, wizards, automatic software generators, etc.), it is reasonable to claim that it is human error that caused the failure of the development environment tool. The causes of software errors can be further classified as follows according to the stages of the software development process in which they occur.

1) Faulty Definition of Requirements: The faulty definition of requirements, usually prepared by the client, is one of the main causes of software errors. The most common errors of this type are:

- Absence of vital requirements.
- Incomplete definition of requirements. For instance, one of the requirements of a municipality's local tax software system refers to discounts granted to various segments of the population: senior citizens, parents of large families, and so forth. Unfortunately, a discount granted to students was not included in the requirements document.

2) Client–Developer Communication Failures: Misunderstandings resulting from defective client–developer communication are additional causes for the errors that prevail in the early stages of the development process:-

- Misunderstanding of the client's instructions as stated in the requirement document.
- Misunderstanding of the client's requirements changes presented to the developer in written form during the development period.
- Misunderstanding of the client's requirements changes presented orally to the developer during the development period.

3) Deliberate Deviations from Software Requirements: In several circumstances, developers may deliberately deviate from the documented requirements, actions that often cause software errors. The errors in these cases are byproducts of the changes. The most common situations of deliberate deviation are:

© 2015, IJARCSSE All Rights Reserved

- The developer reuses software modules taken from an earlier project without sufficient analysis of the changes and adaptations needed to correctly fulfill all the new requirements.
- Due to time or budget pressures, the developer decides to omit part of the required functions in an attempt to cope with these pressures.

4) Logical Design Errors: Software errors can enter the system when the professionals who design the system – systems architects, software engineers, analysts, etc. – formulate the software requirements. Typical errors include:

- Definition that represent software requirements by means of erroneous algorithms.
- Process definitions that contain sequencing errors. For example, the software requirements for a firm's debt-collection system define the debt-collection process as follows. Once a client does not pay his debts, even after receiving three successive notification letters, the details are to be reported to the sales department manager who will decide whether to proceed to the next stage, referral of the client to the legal department. Erroneous definition of boundary conditions. For example, the client's requirements stated that a special discount will be granted to customers who make purchases more than three times in the same month. The analyst erroneously defined the software process to state that the discount would be granted to those who make purchases three times or more in the same month.

5) Coding Errors: A broad range of reasons cause programmers to make coding errors. These include misunderstanding the design documentation, linguistic errors in the programming languages, errors in the application of CASE and other development tools, errors in data selection, and so forth.

6) Non-Compliance with Documentation and Coding Instructions: Almost every development unit has its own documentation and coding standards that define the content, order and format of the documents, and the code created by team members. To support this requirement, the unit develops and publicizes its templates and coding instructions. Members of the development team or unit are required to comply with these requirements. One may ask why non-compliance with these instructions should cause software errors. Team members who need to coordinate their own codes with code modules developed by "non-complying" team members can be expected to encounter more than the usual number of difficulties when trying to understand the software developed by the other team members.

- Individuals replacing the "non-complying" team member (who has retired or been promoted) will find it difficult to fully understand his or her work.
- The design review team will find it more difficult to review a design prepared by a non-complying team.

7) Shortcomings of the Testing Process: Shortcomings of the testing process affect the error rate by leaving a greater number of errors undetected or uncorrected. These shortcomings result from the following causes:

- Incomplete test plans leave untreated portions of the software or the application functions and states of the system.
- Failures to document and report detected errors and faults.
- Failure to promptly correct detected software faults as a result of inappropriate indications of the reasons for the fault.
- Incomplete correction of detected errors due to negligence or time pressures.

8) Procedure Errors: Procedures direct the user with respect to the activities required at each step of the process. They are of special importance in complex software systems where the processing is conducted in several steps, each of which may feed a variety of types of data and allow for examination of the intermediate results. For example, "Eiffel", a chain of construction materials stores, has decided to grant a 5% discount to customers, who are billed once a month. The discount is offered to customers whose total purchases in the last 12 months exceed $1 million.

9) Documentation Errors: The documentation errors that trouble the development and maintenance teams are errors in the design documents and in the documentation integrated in to the body of the software. These errors can cause additional errors in further stages of development and during maintenance. Another type of documentation error, one that affects mainly the users, is an error in the user manuals and in the "help" displays incorporated in the software. Typical errors of this type are:

- Omission of software functions.
- Errors in the explanations and instructions given to users, resulting in "dead ends" or incorrect applications.

B. Software Defect tracking system and their relations with the software quality:

Tracker is very cost-effective. Most installations recover their investment within a few short months through increased productivity, improved products, improved customer satisfaction, and sometimes simply by ensuring each fix of an important bug makes it into the next release.

Software Quality has different aspects that influence the software development process such as: quality control, quality improvement, and quality assurance. The detected problems are then corrected" .There are many software tools that play an important role in tracking defects of software and which are called "Defects Tracking Systems". Jalbert defined them as "Allow users to report, describe, track, classify and comment on bugs reports and feature requests".

© 2015, IJARCSSE All Rights Reserved

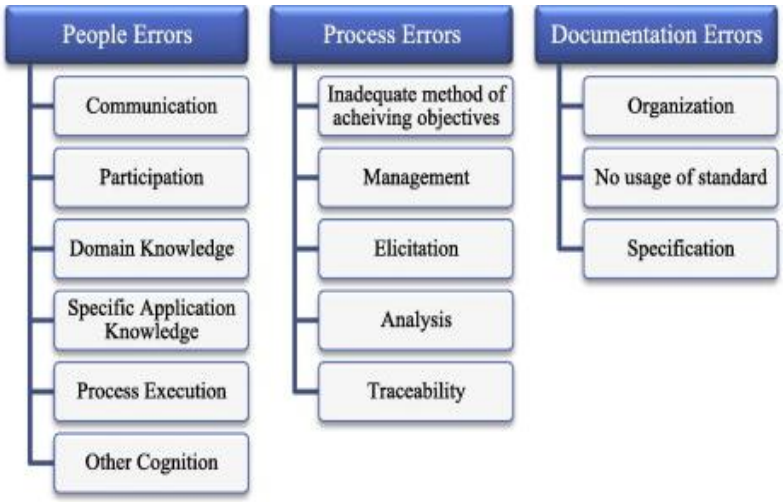

Fig.1 Tracking Issues Manually

Fig.2 Tracking Issues manually in different areas

C. Software Process Improvement Workflow Stages:

Software process improvement (SPI) is viewed as improving the software processes for the intent of increasing the quality of the software products. This can be done through understanding the original software process and change it in order to increase the quality of the software products. Grady claims software defect data is the most valuable source of information for software process improvement decisions. Further, the defect data provides a way of comparing improvements done against historic defect data in order to measure the effect of the improvements. There are five types of the software process improvement workflow stage:

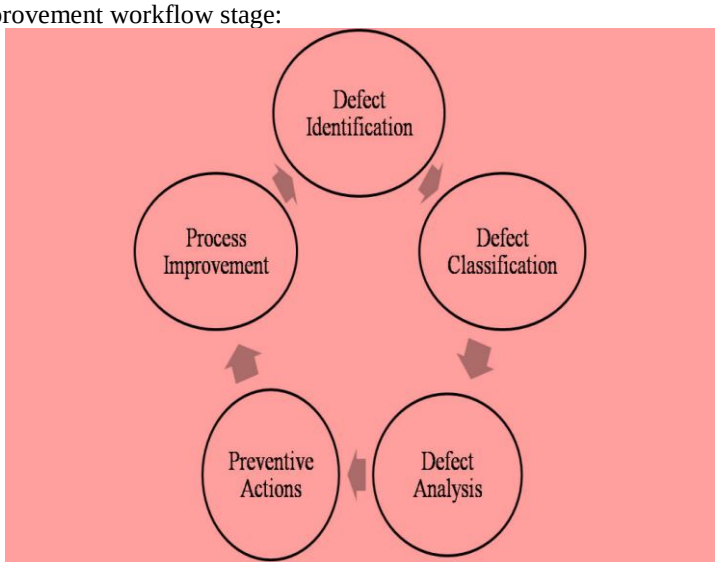

Figure 1.1: Process Improvement Workflow

© 2015, IJARCSSE All Rights Reserved

1.) Defect identification
Defects are found by preplanned activities specifically intended to uncover defects. In general, defects are identified at various stages of software life cycle through activities like Design review, Code Inspection, GUI review, function and unit testing. Once defects are identified they are then classified using first level of Orthogonal Defect Classification.

2.) Defect classification
Orthogonal Defect Classification (ODC) is the most prevailing technique for identifying defects wherein defects are grouped into types rather than considered independently. ODC classifies defect at two different points in time. Time when the defect was first detected – Opener Section. Time when the defect got fixed –Closer Section.

3.) Defect analysis
Defect Analysis is using defects as data for continuous quality improvement. Defect analysis generally seeks to classify defects into categories and identify possible causes in order to direct process improvement efforts. Root Cause Analysis (RCA) has played useful roles in the analysis of software defects. The goal of RCA is to identify the root cause of defects and initiate actions so that the source of defects is eliminated.

4.) Defect prevention
Defect prevention is an important activity in any software project. The purpose of Defect Prevention is to identify the cause of defects and prevent them from recurring. Defect Prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future.

5.) Process improvement
The suggested preventive actions are implemented by rewriting the existing quality manuals and tweaking the SDLC processes and come out with an improved SDLC processes and documents. Next set of projects follow the revised quality processes there by effectively all the preventive actions are followed meticulously.

D. Error Types:
A few programming errors caused most of the overlays.
These are:

1. **Allocation Management**: One module deal locates a region of memory before it has completely finished with it. After the region is reallocated, the original module continues to use it in its original capacity.
2. **Copying Overrun:** The program copies bytes past the end of a buffer.
3. **Pointer Management:** A variable containing the address of data was corrupted. Code using this corrupted address caused bugs in related parts of the code.
4. **Client Code:** A few errors were caused by errors in application code running in protected mode.
5. **Recovery or Error Handling:** Recovery code is notoriously difficult to debug and difficult to test completely. When an error is discovered, the system runs a recovery routine to repair the error. The recovery code could have errors.

II. PROPOSED WORK

The proposed work is about to study and analysis of software defects in a software system for process improvement by using different clustering approaches. The Software Defects is one of the common measure of software quality.
The proposed work is about to study and analysis of software quality in a software system under the consideration of software and the hardware based analysis. The Software Reliability is one of the common measure of software quality. In our proposed work we are presenting to perform the hardware and software based reliability analysis at the first level. If the hardware oriented criticality is feasible then the software based analysis will be performed. For this analysis the fault analysis will be performed in each module. The complete work is divided in 2 main steps.

1. Analysis of Software Defects and Data Collection related to software as well as hardware based defects.
2. Conclude the overall results from the system

III. PROBLEM DEFINITION

As the software development begins there also exists the probability of occurrence of some defect in the software system. Software defects plays important role to take the decision about when the testing will be stopped. Software defects are one of the major factors that can decide the time of software delivery. Not only has the number of defects also the type of defect as well as the criticality of a software defect affected the software quality. Software cannot be presented with software defects. We are trying to categorize the software defects using some clustering approach and then the software defects will be measured in each clustered separately. The proposed system will analyze the software defect respective to the software criticality and its integration with software module. Defect prevention techniques are used for detecting defects from the software so that we improve the quality of the process.

IV. OBJECTIVES

- Study the software Defects and its impact on software system.
- A Survey on uses of study of software defects.

© 2015, IJARCSSE All Rights Reserved

- Collecting the secondary data from some software industry respective to software defects.
- Implementation of Clustering Algorithm with software Defects.
- Analysis of software defects in the system using clustering.

V. CONCLUSION

This Paper described terms and findings from significant earlier research, thereby forming a conceptual context and foundation for the exploratory observational study that is central to this research. The first part was a discussion of different **Classification of the Causes of Software Errors**, on the one hand, defect tracking systems and its relation with the software quality and, on the other hand, different classifications of defects with phases of their tracking and shortcomings of these models.The last section described the various models of the **Software Process Improvement Workflow Stages** of the defect tracking system. It was argued that the previously discussed models concentrated on the tracking phases of defects without caring of the classification factors of the defects. Furthermore, there were different directions of preventing defects and classification of bugs in various models, in chronological order.

ACKNOLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Er. Ritika Asst. Prof. at RPIIT Technical campus karnal, who gave me the golden opportunity to do this wonderful Paper on the topic ***"****A Proposed Defect Tracking System for Classifying of the Causes of Software Errors"*, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this paper within the limited time frame.

REFERENCES

[1] *"Software bugs cost US economy dear"*. Web.archive.org. June 10, 2009. Retrieved September 24, 2012.
[2] Humphrey, W. (1996), "Introduction to Personal Process Improvement", *Addison- Wesley, MA, 1996*
[3] Watts, S.H., (1989). Managing the Software Process, Defect Prevention, *ISBN 0102180952, 9780201180954, Addison Wesley Professional Publisher.*
[4] Creswell, J.(2003), "Research Design Qualitative, Quantitative and mixed Methods Approaches 2nd Edition", *SAGE Publications Inc, USA, 2003*
[5] Deming, WE. (1986), "Statistical Method from point of view of Quality Control," *publisher Dover, Newyork, 1986*.
[6] Manish Mahajan, "Hierarchical Clustering Approach for Modeling of Reusability of Function Oriented Software Components" New York 1990.
[7] P.V.Ingle, M.M.Deshpande (2013) "Software Quality Analysis with Clustering Method" *International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868 Foundation of Computer Science FCS, New York, USA.*
[8] Leszak, M., Perry, E.D., Stoll, D., (2000). A Case Study in Root Cause Defect Analysis, *Proceedings of the 22nd International Conference on Software*
[9] Vasudevan, S., (2005). Defect Prevention Techniques and Practices, *Proceedings of fifth annual International Software Testing Conference,* Banglore, India
[10] Nair T.R., (1993) "Four-Step Approach Model of Inspection (FAMI) for Effective Defect Management in Software Development", Bangalore, India.

© 2015, IJARCSSE All Rights Reserved