I.J.Modern Education and Computer Science, **2012, 3, 14-20**
Published Online April 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2012.03.02

Contemporary Trends in Defect Prevention: A Survey Report

Muhammad Faizan
Department of Computing,
SZABIST, Islamabad, Pakistan
damsonsslad@yahoo.com

Muhammad Naeem Ahmed Khan
Department of Computing,
SZABIST, Islamabad, Pakistan
mnak2010@gmail.com

Sami Ulhaq
Department of Computing,
SZABIST, Islamabad, Pakistan
uomiansame007@gmail.com

***Abstract*—Most of the software projects fail to meet the desired level of quality and standards due to different types of defects introduced during the course of requirement solicitation, designing and development. These defects inexorably hinder the secure deployment or smooth operations of the software systems. One of the key reasons for this misfortune is the lack of proper defect prevention planning while formulating the software architecture. Defect prevention needs to be a thorough and critical phase because it has a direct impact on quality of the product which cannot be compromised. This paper looks into different defect prevention techniques and analyses them critically. The scope of this study is restricted to the identification of the modern trends in defect prevention.**

***Index Terms*—Defect Prevention, Causal Analysis, Root Cause Analysis, Orthogonal Defect Prevention, Action Based Defect Prevention.**

1. INTRODUCTION

Defect prevention is a quality assurance alternative for improving quality of software product. The focus of defect prevention techniques is to produce better quality products within the available budget and time. Quality of a product directly relates to the number of defects - minimal defects lead to achieve better quality. Most of the software projects fail to meet the desired level of user acceptance and standard of quality as they do not adequately employ effective defect prevention strategies.

Defect prevention techniques by and large are applied in many fields during production and development phases, however this study confines to the subject of software defect prevention. Defect prevention is normally performed at every stage of the software development lifecycle (SDLC). The major activities related to software defect prevention include: creating software defect prevention plan, defect inspection, defect causal analysis, defect classification, defect prioritization and defect detection. Suma and Nair [1] consider that defect pertains to any inaccuracy or blemish in the software product or process. Generally, the term defect refers to an error, bug, fault or failure [2]. The terms error, fault and failure can further be elaborated as ‘action that thrust to incorrect result’, ‘erroneous decision taken due to wrong interpretation of the available information’ and ‘lack of ability to meet the expected performance’ respectively. Defect prevention techniques aim at prohibiting defects to occur beforehand [3]. Alternatively, defect prevention can be defined as a quality improvement process by identifying defects through root causes analysis and formulating effective corrective and preventive measures to inhibit these defects from recurring in future [1,4].

Similarly, casual analysis is used for Root Cause Analysis (RCA) [5, 3] which is the prime technique for identification of risks associated with distinctive types of defects. Orthogonal Defects Classification (ODC), devised by IBM, is a commonly used technique for unearthing and classifying defects in the software products [6, 7, 8]. Bean [9] recommends PACRAT (Performance and Continuous Re-Commissioning Analysis Tool) for creation, revision and archiving during defect prevention process. Due to diversified nature of the software products, there is no perfect and ideal solution to preclude defects.

This paper is organized into four sections. This section being the first section provides a brief introduction of the defect prevention and its classification. The second section summarizes the related work along with a critical analysis of the techniques described in the literature review. The perspective future work and conclusion are described in the third and fourth sections respectively.

2. LITERATURE REVIEW AND CRITICAL ANALYSIS

Suma and Nair [1] describe the importance of defect prevention approaches by highlighting that software inspection that requires 15% of total project time and is categorized as the fruitful quality aspect of a product as it reduces development cost and time. Defect removal efficiency ratio and orthogonal defect classification techniques are also valuable for the prevention of defects. Through training and applying well-known standards, an organization can thrive for a better environment and improvement in quality standards. The strength of the study is that it provides the comparison among different approaches and techniques of defect prevention. Meticulous analysis of different organizational levels analysis is enunciated.

Chang *et al.* [3] stress on the importance of take preventive measures to control the defects in the early stages of the project. Causal analysis is a commonly used technique to ascertain the causes of defects and take remedial actions. However, when the reported defects grow in numbers, it becomes too difficult to handle and take preventive actions. The defect prediction is very important for defect prevention and software process improvement. The study presents defect prediction approach based on association rules, which applies association mining technique. The proposed approach can be applied to the software development process to predict the actions that are likely to cause high defect rates. (ABDP approach coupled with classification decision tree technique is used to build a prediction model, and performs association rule mining on the defects. The association rule mining is data mining technique used to find frequent patterns, associations, correlations or causal structures among a large dataset. For the validation, the proposed approach is applied to a business project.

Kumaresh and Baskaran [4] classify the defect data using ODC and defect analysis methodology. On the basis of defect analysis, root cause analysis is conducted using cause-effect diagram and preventive actions are implemented. Through this methodology, defects density is reduced and productivity is improved. While using ODC methodology, it is observed that most of the defects are related to logical error (coding) and requirements. The results show that defect prevention plays a vital part in every project lifecycle and reduce the defect reoccurrence density which leads to reduction in rework, cost and effort. Defect density is measured in terms of:

Defect Density = # of defects / size (KLOC) … (1)

The study discusses the defect prevention phase as a whole for the entire project lifecycle by exhaustively analyzing the organizational level data.

Marcos *et al.* [5] describe defect casual analysis implementation for the process improvement in the organization. By applying the DCA concept, study shows that the reduction of defect rate still remains at 50% in different organizations. To achieve better quality of products, the following six steps are recommended to be implemented through the DCA:

1. Sample of the defects
2. Classifying defects
3. Summarizing repeated defects
4. Defining causes
5. Action planning
6. Documentation

The study also introduces the approach of DBPI (defect prevention based process improvement), which seems to be a promising technique to improve the quality. Furthermore, the study also highlights that Pareto and cause-effect techniques are more appropriate to implement DCA.

Study introduced DBPI and stresses that Pareto and cause-effect technique are commonly used for implementation of DCA.

Trivedi and Pachori [6] focus ODC methodology to implement defect prevention in software industry. Real world implementation of ODC provides better way to classify and detect defects. HP and IBM use ODC to classify software defects and find out the root causes for preventive action. By applying ODC methodology, defect attributes are identified as defect trigger, defect type and defect qualifier. Results are analyzed after implementation of ODC in software development lifecycle which shows that defect detection and cost of defect detection is reduced.

The contribution of the study is that it describes the importance of ODC in project life cycle and how it introduces the quality in product. How the improvement of processes are conducted and evaluated.

Shenvi [7] highlights the importance of defect prevention using ODC to classify the defects. Defect prevention improves the quality of product and also reduces the cost and development time of the project. After classifying defects, root cause analysis is done to prevent them occurring again. Orthogonal methodology classifies every defect into mutually exclusive attributes of technical and managerial types. Case study of real-life software development project is presented in this paper to show how this methodology fits together in the existing process framework. Defect prevention workflow is discussed which include steps like defect identification, defect classification, casual analysis, tracking, measurement, results verification and learning's. After applying root cause analysis, some causes appeared as missing requirements and some incorrect requirements.

Defect prevention work flow is one the best practice. CMMI key process area is presented. Importance of defect prevention is described using the real life DVD project. Implementation of DOORS tool is discussed.

Tiejun *et al.* [8] define a defect tracing system as a new defect analytical methodology of defect prevention technique which is totally based on ODC. The study presents workflow of the defect tracing system and shows how this methodology improves the accuracy of identifying defects. ODC reference model is Omni-directional analysis and has four basic attributes: Reason, Consequence, Verification and development. By using the work flow activities and ODC concept this bug tracking system is proposed, through using this technology bugs can easily traced. Bug tracing system (BTS) first initializes the process through submitting the problem which record all attributes related to the bug and then manager processes the verification and follow on activities defined in workflow of BTS.

By applying BTS methodology, the defect processing has improved to 40% and management is able to make a quality evaluation on every software module.

Bean [9] suggests implementation of defect prevention and defect detection technique in the test application development tool named Performance and Continuous Re-Commissioning Analysis Tool (PACRAT). The PACRAT and PT3300 are used with combination. Defect prevention activities are implemented to build the automated test equipment like tracking defects, identifying the root causes of defects and providing solution to these issues before the development of the tool. A database has been created for the validation and syntax check including the entire defect detection activities. PCRAT is one of the best automated tools regarding implementation of defect prevention and defect detection activities. Tools are explained thoroughly regarding standards like CMMI and SIX SIGMA.

Meng *et al.* [10] Proposed a defect prevention framework that decreased defect density from 0.85 to 0.1 per kilo line of code (KLOC). Project planning and process tracking are defined by using the data analysis technique. Defect prevention activities are carried out in different phases of project life cycle to achieve better quality level. The defect prevention activities include: *Preparation, Definition of Defect Type, Defect Prevention Process* and *Support for Defect Prevention.* The splitting of defect types is presented using the Histogram and Pareto diagrams. The study shows that the defect prevention activities bring product quality improvement by removal of the defect density and better environment in the organization for the project stakeholders.

Suma and Nair [11] focus on inspection technique of defect prevention and detection by carrying out inspection on five dissimilar projects of different sizes. The study reports that 70% of defects are uncovered during the inspection and developer unit testing process while 29% defects are detected during the validation phase. These statistics are helpful to improve confidence level in terms of quality of the product. The inspection yield for a project is calculated as a percentage named

Defect Removal Efficiency by using the following formula:

$$\frac{\text{total number of defects estimated}}{\text{actual number of defects identified}} \times 100 \quad (2)$$

The study highlighted the utility of inspection as a solution for defect prevention and detection.

Chang and Chu [12] propose an action based defect prevention (ADBP) for the implementation of defect prevention in the software development process. ABDP includes activities like first classifying the actions which becomes the reason for the defect introduction in the system. Dataset is collected and feature subset selection technique is applied on the dataset which filters out attributes which are not useful in the dataset. Data sampling is further applied to sample the major classes of dataset using the under-sampling technique. After analysis phase, the prediction model is generated which is helpful for the defects blocking. The data set of AMS-COMFT project is presented and ABDP approach is applied in which accuracy of defect based actions are predicted.

Mujtaba *et al.* [13] discuss software quality metrics and propose Defect Fixation Efficiency (DFE) technique that quantitatively measures the defect fixation process. The defect fixation is calculated by using the formula:

$$DFE = \frac{\text{Total number of defects fixed}}{\text{Total no defects} + \text{New injected defects}} \quad (3)$$

The higher DFE percentage indicate high number of defects A 2×2 matrix is defined after calculating the DFE, to indicate quality of the software. The matrix encompasses four scenarios: High-High, High-Low, Low-High and Low-Low to reflect the relationship of DFE to the wrong defects and shows the customer satisfaction with the product's quality.

A new software quality assurance metric is defined to show relationship among software quality attributes.

Gerard *et al.* [14] describe that *simplicity* and *reliability* are the two main factors to prevent the failure of the application. Reliability is achieved through implementation of two strategies - one of them includes simplicity and sturdiness and the second is takes advantage of idleness. Through probability techniques, importance of individual and whole system failure is described.

Ghazarian [15] conducted a case study on defect introduction mechanisms and highlights the importance of causal analysis. Case study is conducted on three ERP software systems and shows that external factors including incomplete requirements specifications, adopting new unfamiliar technologies, lack of requirements traceability and the lack of proactive and explicit definition and enforcement of user interface consistency causes 59% of the defects.

Jantti *et al.* [16] argue that a well-organized defect management process is one of the success factors for implementing software projects in time and in budget. The defect management process activities are defect prevention, defect discovery and resolution, defect causal analysis and the process improvement. However, establishing a defect management process is not much easier. The reported problems are related to defect resolution reports, limited project resources to fix defects and challenges in creating a test environment.

A summary of critical analysis of different defect prevention techniques is provided in Table I.

Table 1: Critical Analysis of Defect Prevention Techniques.

Ref#	Method/Technique	Strength	Considerations/Scope
[1]	Inspection	The strength of the study is that it provides the comparison among different approaches and techniques of defect prevention. Meticulous analysis of different organizational levels analysis is enunciated.	The author did not discuss any software tool to implement the defect prevention technique.
[3]	Action Based Defect Prediction and Association Rule Mining	The combination of ABDP and association rule technique provides better results for software process improvement.	The proposed technique can only be used when the numbers of defects are larger.

Ref#	Method/Technique	Strength	Considerations/Scope
[4]	ODC & RCA	The study discusses the defect prevention phase as a whole for the entire project lifecycle by exhaustively analyzing the organizational level data.	The paper did not discuss about complex size projects for the root cause analysis. Comparison of techniques is not elaborated.
[5]	Defect Causal Analysis (DCA)	Study introduced DBPI and stresses that Pareto and cause-effect technique are commonly used for implementation of DCA.	The study is simply restricted to the use of Pareto and cause-effect techniques and no comparison among different techniques is drawn.
[6]	ODC	The contribution of the study is that it describes the importance of ODC in project life cycle and how it introduces the quality in product. How the improvement of processes are conducted and evaluated.	The study only focuses on only one area.
[7]	ODC & RCA	Defect prevention work flow is one the best practice. CMMI key process area is presented. Importance of defect prevention is described using the real life DVD project. Implementation of DOORS tool is discussed.	The study lacks suggesting measures to further reduce improve the defect rates.
[8]	Bug/defect tracing system (BTS)	By applying BTS methodology, the defect processing has improved to 40% and management is able to make a quality evaluation on every software module.	The study is merely based on a case study.
[9]	Performance and Continuous Re-Commissioning Analysis Tool (PACRAT)	PCRAT is one of the best automated tools regarding implementation of defect prevention and defect detection activities. Tools are explained thoroughly regarding standards like CMMI and SIX SIGMA.	The paper did not discuss about the development of the complex automated software tools and not related to the project life cycle.
[10]	Defect prevention process improvement	The study shows that the defect prevention activities bring product quality improvement by removal of the defect density and better environment in the organization for the project stakeholders.	Paper did not discuss about the project complexity level regarding implementing the defect prevention activities.
[11]	Inspection	The study highlighted the utility of inspection as a solution for defect prevention and detection.	The study is conducted truly in generic form and lacks description about tools for software inspection.
[12]	Action Based Defect Prevention	ABDP approach can improve software processes as it consists of different techniques and technologies.	Work nature of the technique is too complex; however, it is useful when numbers of defects are large.
[13]	SQA Matrix, Defect Fixation Efficiency	A new software quality assurance metric is defined to show relationship among software quality attributes.	The study did not draw comparison of DFE with other techniques.
[14]	Simplicity & reliability	A novel method for defect prevention using backup module is proposed in the study.	The paper did not include drawbacks of applying this technique and no implementation details are provided.
[15]	Case study	This study highlights the key areas where effort should be directed.	The study has limited scope as it is conducted only on ERP systems.
[16]	Case Study	Major problems are identified in defect management process which can help organizations to identify and avoid defects in the products.	The study is limited to the managerial aspects for handling defects.

3. FUTURE WORK

As a prospective future work to this research, we intend to propose a framework for defect prevention process that is helpful to enhance quality and minimize the impact of defects.

4. CONCLUSION

Defects are a common phenomenon to software products which require more focused attention to be paid to prevent defects from occurring in advance because at early stages if defects are prevented it will require less effort and budget to fix them. In this paper, a critical analysis of different defect prevention techniques is presented. The study particularly takes into account the effectiveness of defect prevention for software quality improvement. It is observed that inspection is a common, straightforward and an effective defect prevention technique; and for identification of defects, causal analysis is the most frequently used approach. However, new defect prevention techniques that emphasize on the classification of defects are getting popularity like ODC ABDP.

REFERENCES

[1] V. Suma and T. R. G. K. Nair, "Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels," *World Academy of Science, Engineering and Technology (WASET)*, 42, pp. 258_262, 2008.

[2] B. Clark and D. Zubrow, How Good Is the Software: A review of Defect Prediction Techniques, sponsored by the U.S. department of Defense by Carnegie Mellon University, version 1.0, page 5, 2001.

[3] C. P. Chang and C. P. Chu., Defect prevention in software processes: An action based approach, The Journal of Systems and Software 80, 559–570, 2007.

[4] S. Kumaresh and R. Baskaran, Defect Analysis and Prevention for Software Process Quality Improvement, International Journal of Computer Applications, Volume 8– No.7, pp. 42_47, 2010.

[5] M. Kalinoski, G. H. Travassos and D. N. Card., "Towards a Defect Prevention Based Process Improvement Approach," 34th Euromicro Conference Software Engineering and Advanced Applications (SEAA), IEEE Computer Society, pp. 199_206, 2010.

[6] P. Trivedi. & S. Pachori, Modeling and Analysis of Software Defect Prevention Using ODC, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 1, No. 3, pp. 75_77, 2010.

[7] A. Shenvi, "Defect Prevention with Orthogonal Defect Classification," International Software Engineering Conference (ISEC'09), ACM, pp. 83_87, 2009.

[8] P. Tiejun, Z. Leina and F. C. bin., "Defect Tracing System Based on Orthogonal Defect Classification," International Conference on Computer Science and Software Engineering (CSSE), IEEE, pp. 574_577, 2008.

[9] E. Bean, Defect Prevention and Detection in Software for Automated Test Equipment," Instrumentation & Measurement Magazine, IEEE, Volume: 11 Issue: 4, pp. 16_23, 2008.

[10] M. Li, H. Xiaoyuan and A. Sontakke, "Defect Prevention: A General Framework and Its Application", Proceedings of the sixth International Conference on Quality Software (QSIC'06), IEEE Computer Society, 2006.

[11] V. Suma and T. R. G. K. Nair, "Enhanced Approaches in Defect Detection and Prevention Strategies in Small and Medium Scale Industries," Third International Conference on Software Engineering Advances (ICSEA), IEEE Computer Society, pp. 389_393, 2008.

[12] C. P. Chang and C. P. Chu, Defect prevention in software processes: An action based approach, The Journal of Systems and Software 80,559–570, 2007.

[13] G. Mujtaba, Dr. T. Mahmood and Professor Z. Nasir, A Holistic Approach to Software Defect Analysis and Management, Australian Journal of Basic and Applied Sciences, 5(6), 1632_1640, 2011.

[14] G. J. Holzmann and R. Joshi, Reliable Software Systems Design: Defect Prevention, Detection, and Containment, International Federation for Information Processing (IFIP). LNCS 4171, pp. 237_244, 2008.

[15] A. Ghazarian, "A Case Study of Defect Introduction Mechanisms", Conference on Advanced information Systems Engineering (CAiSE), LNCS 5565, pp.156–170, Springer Verlag Berlin Heidelberg, (2009).

[16] M. Jantti, T. Toroi and A. Eerola, "Difficulties in Establishing a Defect Management Process: A Case Study", International Conference on Product Focused Software Development And Process Improvement (PROFES'06), LNCS 4034, pp.142–150, Springer Verlag Berlin Heidelberg, (2006).

Authors

Muhammad Faizan is currently pursuing MS in Software Engineering at SZABIST, Islamabad,

Pakistan. His research areas include Software Defect Prevention and Software Quality Engineering.

Muhammad Naeem Ahmed Khan obtained D.Phil. degree in Computer System Engineering from the University of Sussex, Brighton, England, UK. Presently, he is affiliated with Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad. His research interests are in the fields of software engineering, cyber administration, digital forensic analysis and machine learning techniques.

Sami Ulhaq is a student of Masters in Software Engineering at SZABIST Islamabad, Pakistan. His focused research areas are Software Quality Engineering, Software Requirement Engineering and Global Software Development.

