I.J. Information Engineering and Electronic Business, 2014, 1, 32-41
Published Online February 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2014.01.04

A Survey on Effective Defect Prevention - 3T Approach

Priyanka Chandani
JSS Academy of Technical Education, Noida
priyanka.chandani@hotmail.com

Chetna Gupta
Jaypee Institute of Information Technology, Noida
chetnagupta04@gmail.com

Abstract — Defects are most detrimental entities which deter the smooth operation and deployment of the software system and can arise in any part of the life cycle, they are most feared, but still Defect Prevention is mostly discounted field of software quality. Unattended defects cause a lot of rework and waste of effort. Hence only finding the defects is not important, finding the root cause of the defect is also important which is quite difficult due to levels of abstraction in terms of people, process, complexity, environment and other factors. Through this study various techniques of Defect classification, prevention and root cause analysis are analysed. The intent of this paper is to demonstrate the structured process showing defect prevention flow and inferring three T's (Tracking, Technique and Training) after analysis.

Index Terms — Defect, Defect Analysis, Defect Prevention, Orthogonal Defect classification, Root cause analysis.

1. Introduction

It is better to question than to worry later" holds as much importance in software development and quality conformation as in English literature. The illness of software development can truly be healed by concrete verification and validation. The requirement breakdown and analysis from the stakeholders of the engagement is very important and upon stages of software life cycle, they develop into a concrete product. Failing to produce a viable product, can make the customer unhappy and defeat the objective. So, it is quintessential to capture problems (defects) in all stages to ensure zero residual nature of the product and improved quality. Quality comprises of all characteristics and features of a product which refers to satisfy a given requirement. It should conform to specifications, customer satisfaction, and value of the product, people, service, and processes. Many companies that produce software have Software Quality Assurance departments, designed to ensure software quality where the focus is on Defect Prevention [1].

A defect is a variance from a desired behaviour, which affects the quality of the software. Defect Prevention identifies those defects, correct them and prevent them from reoccurring. The aim of defect prevention is to produce good quality products within the budget and time. We know no software can be built as Defect Free; defects may be introduced during specification, design, and coding of the test application. Hence, defect prevention is an essential part of the software process quality improvement, which cannot be compromised [2][3]. Defect analysis and prevention techniques have been applied successfully in a number of software development organizations with significant reductions in errors [4][5][6].

A technique which emphasise on identifying the root causes of defects and initiate the action to correct them is Root cause Analysis which looks simple, but is very deep rooted. Root cause analysis for defects is proved to be a successful process in defect prevention [7][8][9]. Defects are analysed one by one qualitatively which is very time consuming and rigorous process. Hence we need to view them in collections and should be able to find the causes at the right level. Various classifications have been developed over the last decades for improving defect detection or educating developers [10][11][12]. Orthogonal Defects Classification (ODC) [12], devised by IBM, is commonly used technique for classifying defects in the software products. Defects are identified and analysed for patterns to improve the quality of the software process. It provides a meaningful classification of defects into classes that collectively point to the process that needs attention [12]. Various case studies have shown that ODC can improve the effectiveness and efficiency of development and testing which is important for quality improvements [13][14][15][16]. In this paper, an attempt is made to present a comprehensive view of defect prevention techniques and analyses them critically by stating the advantages and limitations of selected approaches and inferring 3T's of Defect Prevention through this study.

The remaining paper is organized as follows: In the next section, data extraction from different sources showing distribution of selected papers is presented. An overview of related work on defect prevention along with the critical analysis of the techniques is presented in the third section. The defect Prevention flow and

three pillars of Defect Prevention along with learnings from this study are described in fourth and fifth section respectively and finally a sixth section presents the conclusion.

2. Data extraction

A systematic literature review is being done after searching widely on electronic database [17]. A total of 56 papers was founded; 40 were selected, and some basic information was extracted. In this section, distribution of papers is shown based on year, source and technique used.

These papers are distributed in different categories as shown below in Fig 1 based on technique or approach used in the paper.

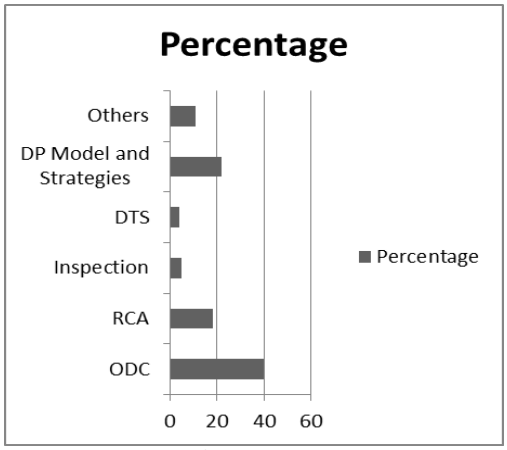

Figure 1: Distribution of selected papers based on technique

Fig 2 shows the distribution of the selected papers by years and Table 1 shows the publication venues of selected papers.

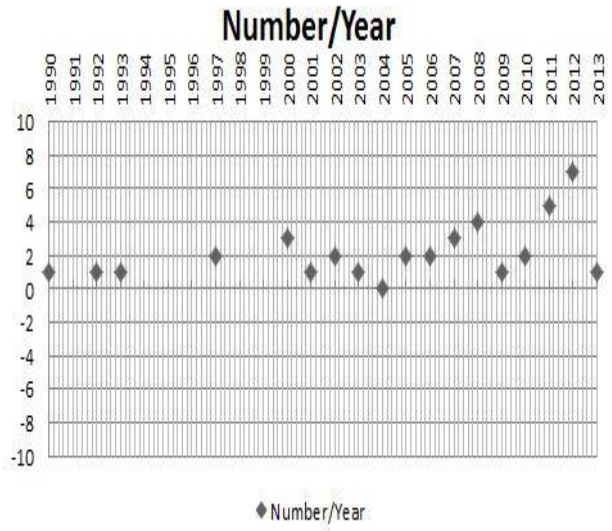

Figure 2: Distribution of the selected papers in different years

TABLE 1: Selected papers across different venue

Type	Description	Total
Journal	IEEE Transactions on Software Engineering	1
	IEEE Software	4
	Journal of Systems and Software	3
	IEEE Journal on Selected Areas in Communications	1
	The Journal of Defence Software Engineering	1
	Journal of Information and Software Technology	1
	International Journal of Computer Science and Management Studies	1
	IBM Systems Journal	1
	International Journal of Computer Applications	1
	International Journal of Advanced Computer Science and Applications	1
	International Journal of Software Engineering & Applications	2
	International Journal of Computer Technology and Application	1
	European Journal of scientific Research	1
	International Journal of Future Computer and Communication	1
	World Academy of Science, Engineering and Technology	1
Conference	India Software Engineering Conference	1
	International Conference on Environmental Science and Information application Technology	1
	International Conference on Information and Communications Technology	1
	International Conference on Automated Software Engineering	1
	International Conference on Software Maintenance	1
	International conference on software quality	1
	International Conference on Computer Science and Software Engineering	1
	International Conference on Quality Software	1

	International Conference on Applications of Software Measurement	1
	India Software Engineering Conference	1
	International Conference on Software Engineering	1
	International Software Metrics Symposium	1
Others	International Symposium on Empirical Software Engineering	1
	ACM SIGSOFT Software Engineering Notes	2
	Instrumentation & Measurement Magazine, IEEE	1
	Workshop on Defects in large software systems	1
	Scientific & Academic Publishing	1
	Computing Research Repository	1

3. Related Work

Different analysis strategies to improve software processes in defect prevention have been studied and researched upon. Some of them have been discussed below:

Jalote, Munshi and Probsting [18] discussed When-Who-How strategy of analysis of defect data to improve the software quality process. Here the focus is on the three dimensions- time, technique or process and the group which finds the defects. Improvement is shown when these three dimensions are combined together and applied for the analysis of defect data for one version of windows. The analysis was done on the component level, which showed a correlation between early and late defect density rate and monitored the defect density at various milestones.

Li, Stalharse, Conradi and Kristiansen [19] worked on enhancing the Defect Tracking System (DTS) to improve the quality of the software. The author examined various defect attributes of different tracking system used by ten companies and found that different companies use different defect attributes in DTS, but they were not complete for software process improvement. The authors selected two companies and added some new attributes for these company's DTS like Effort, Fixing types, Triggers, Root cause etc. The improved DTS provided valuable input to developers and testers and thereby reducing the time spent on fixing defects, identifying the root causes, preventing defects well in advance.

Mittal, Solanki and Saroha [20] mentioned that no Software can be "Defect free", after testing defects should be reported using 'Defect tracking system' which can be managed to improve the quality of the software. This paper focused on defect management process and the approach for handling the defects which includes counting and managing the defects, maintaining defect leakage metrics. Control charts were used to measure and improve the processes. Various approaches were discussed to handle the defects like Firefighters, Reactive and Proactive approach.

Hafiz Ansar Khan [21] proposed a defect management process model after viewing the major challenges of ITIL defect management process which is used by most of the organizations. The author applied the proposed model in one of the case organization and found that this model is very easy to use and strengthens the defect management and review process of the organization. The proposed model also observes the defect management from the customer's viewpoint.

Card [22] discussed two approaches to measure and model software quality throughout the life cycle of the software. Empirical and analytical approaches were used to build up defect profile which when studied makes the quality visible to be managed. These approaches were applied and modelled in some real time industry projects, generating defect profile and analysing the departures from the defect profile early in the life cycle. This process can provide feedback to the developers and testers to work on the discrepancies.

Kalinowski, Card and Travassos [7] discussed how to implement a Defect Causal Analysis (DCA) efficiently. This paper discussed the readiness of an organization for DCA. For DCA implementation, Pareto charts and cause effect diagrams are very helpful to identify the causes of defects. Indicators that support DCA were also discussed like number of defects found by size unit, the mean number of defects found per hour of inspection, phase input quality, phase output quality which help identify the efficiency of the DP activities. It was shown that up to 50 percent of improvement in the defect rates can be achieved, however, implementation cost varies from 0.5 to 1.5 percent which is very marginal in comparison to the gain.

Lehtimen, Mantyla and Vanhanen [23] introduced lightweight root cause analysis method (ARCA) to detect the causes and how the corrective actions are taken. Unlike the normal RCA methods, this one does not require heavy start-up investment and can be easily applied in small-medium sized companies, unlike normal Root Cause Analysis (RCA) methods. The author evaluated this approach through field studies at four software companies through feedbacks using interviews, meetings and query forms. This method is easy to use, highly adaptable, feasible and looks very promising for defect prevention.

According to Dalal and Chhillar [8], Root cause analysis was done for some software failures that happened in the past and ongoing software projects. Various RCA methods and processes which help to reduce the chances of software failure were also discussed like Cause-Effect Analysis, Events and Causal Factor Analysis, Fault Tree Analysis, Causal Factor Charting, Brain Storming and 5 Whys. Based on the empirical study of root cause analysis of software failures, it was perceived that lots of software failed at the time of upgrading the software and due to

inadequate system and integration testing [16]. Many good examples of efforts on causal analysis have been published [44].

Navid Hashemi Taba and Siew Hock Ow [24] highlighted some traditional inspections approaches used for more than a decade which are not effective for the current processes. A comprehensive software inspection model was introduced in this paper which performs defect removal activities as an important task of inspection. This proposed model suggested a defect management approach for removing defects iteratively. Customized evaluations of the process prepare important information about the effectiveness of the inspection process. In a real environment, it helps to detect and remove defects.

Suma and Nair [5] discussed the importance of the effective defect prevention approach in software processes highlighting that on an average 15% of inspection and 30% of testing is required for 99% of defect elimination. Author highlighted that the inspection is most valuable and competent technique [5]. A lot of information was provided on various methods and practices for defect prevention and defect detection adopted in five projects.

Ajit Ashok Shenvi [13] worked on presenting a defect prevention framework using Orthogonal Defect Classification methodologies. The structured process for defect prevention mechanism using ODC attributes for defect classification is discussed along with related interpretations for causal analysis and planning. The suitability of this methodology in some real time project was also presented in the paper.

Trivedi and Pachori [14] discussed that various defect measurement and defect tracking mechanism are used for measuring the software quality. In software development life cycle, 40% or the more of the time is utilized on defect detection tasks. This paper focused on the ODC implementation in real world application. It explained about various defect classification schemes and how we can adopt ODC in development of software. This paper mentioned various improvements in software project after implementing ODC.

According to Sakthi and Baskaran [6], the cost of finding and fixing defects is one of the most expensive software development activities, hence better methodologies have to be applied to defect prevention process. Five projects were selected, and different types of defects were first identified, classified using Orthogonal Defect Classification and analysed for patterns to improve the quality of software processes. Defect prevention methods were established for reducing these patterns of similar defects in future projects thereby improving the quality of the projects.

LI and HOU [15] described ODC, analysed the classification of its attributes and effectiveness of software process by the ODC measure method. This study helped in understanding how ODC provides a new software process measurement to improve the software development process and provides a reasonable quantitative standard. It is important how to choose the ODC measure mode corresponding to the different measure objectives to evaluate the software development process and software quality.

Bridge [16] showed how ODC can be used to provide process enhancement feedback to developers also how to measure the progress of development and to emphasise improvement activities where the customer is most impacted. The principals behind ODC were shown by the results of feasibility studies. ODC forms an excellent foundation for the development of defect prevention and quantitative process management techniques [16].

Wagner [10] summarized the work on defect classification approaches that have been proposed by two IT companies IBM and HP. The IBM approach is Orthogonal Defect classification. In IBM, a defect is classified based on defect type, source, impact, trigger, phase found and severity. In HP, defects are classified across three dimensions – Defect origin; types and modes [10].

Kumaresh and Bhaskaran [3] proposed defect framework highlighting the 5 Dimensions (D's) of the defect origin. Each one of the D's concentrates on defects in one particular stage of the software development Lifecycle like Deficiency in Requirements, Design Flaws, Defective Coding Process, Delinquency in Testing and Duration Slippage. Analysis of various defect types was done and the most prominent defects were identified. For each one of the defect type, the reason for such defect was found out, and the Defect Prevention actions were suggested. The author also proposed a defect injection metric based on the severity of the defect instead of defect count. The defect injection metric value, once calculated, serves as a standard to make improvement in the software process development among similar kind of projects.

Langari and Pidduck [25] proposed a new approach by merging Cleanroom methodologies and formal methods for software quality. Cleanroom highlights defect prevention rather than defect removal and Formal methods use mathematical and logical formalizations to find defects early in SDLC [25].

The techniques identified (T1–T13) are mapped to different publications as shown in Table 2. Table 3 summarizes the strength and limitations of various techniques.

TABLE 2: Mapping of Techniques with Publications

Technique		Publication (Reference No)	Key Points
No	Identified		
1.T1	ODC	[6],[12],[13],[14],[15][16],[21],[26],[27],[28],[43],[47],[48]	Defect type, Defect type distribution, Defect Triggers, Principal Association Table, Cause Trigger table
2.T2	RCA	[7],[9],[21],[29]	Defects, causal analysis
T3	Inspection	[5],[30]	Case study, Defect removal efficiency, Testing, Comparative analysis
T4	Comprehensive Inspection Model	[24]	Inspection, Defect Plan, Inspection routines, Evaluation
T5	Defect Origin, Types and Modes	[10]	Origin ,Types, Modes
T6	Defect Tracking System (DTS)	[19],[31],[32]	Defect attributes, SQA, SPI, case studies, metrics and report
T7	Defect Analysis Feedback	[34]	DP team, meetings, trainings, goals
T8	Lightweight Root Cause analysis (ARCA)	[23]	Target problem detection, root cause detection, corrective action innovation and documentation of results , field study
T9	Who-When-How Approach	[18]	Time, technique, team analysis, component level analysis
T10	Bug tracking and reliability assessment system	[35]	Bug tracking system, Comparative analysis, classification
T11	Action Based Defect Prevention	[36]	Feature subset selection, sampling
T12	Empirical and Analytical Model	[22]	Defect Profile, Rayleigh dispersion curve
T13	Cleanroom Methodology	[25]	Cleanroom, Formal methods

TABLE 3: Strength and limitations of various techniques

Collectively point to the process that needs attention. It classifies the defects and help deducing a pattern	T1	Empirical knowledge is less. Different set of dimensions and artifacts.
It analysis the defects one by one to find the root cause of the problem. It is very deep rooted.	T2	This approach is qualitative and labour intensive
Inspection is proved to be most successful technique for defect prevention and detection	T3	There can be some models or tools of inspection which could give better results
This intelligent model gives a defect management approach for removing defects	T4	It is an intelligent model however trainings has to be given to stakeholders to be able to use it correctly
Relationship between defects and document types can be analysed	T5	The triggers in ODC are not directly documented (if compared to ODC)
DTS provides valuable input to developers and testers and reduces the time spent on fixing defects, finding the root causes, preventing defects	T6	Requires lot of motivation and training to the users so that DTS can be used properly. Other problems are incomplete and consistent data or mixed data
It uses iterative development process where defect data from one iteration is used in future iterations in defect prevention	T7	Results might not be as effective with other process models

Unlike the normal RCA methods, this one does not require heavy startup investment and can be easily applied in small-medium sized companies unlike normal Root Cause Analysis (RCA) methods.	T8	The conclusions are made based on personal experience of the case attendees which might affect the results on comparison between ARCA and RCA
Three dimensions combined together can be applied for the detailed analysis of defect data	T9	More work is required to be done in this respect to determine rules of evaluation
It is a automated bug-tracking tool which streamlines the process of reporting, managing and fixing issues	T10	This tool can be incorporated the visualization and machine learning techniques due to which bug assignment can be done automatically during reporting of the bug.
This technique helps in identifying actions that causes defects using FSS and sampling techniques	T11	It is concluded that patterns exist among actions causing many defects. Scope is to apply sequence pattern analysis techniques to increase the prediction performance and identify pattern
These two approaches model and measure the quality throughout the SDLC by building up the defect profile. It gives feedback to the developers and testers	T12	Analytical model are useful when organisation lacks complete life cycle defect data or like to smooth existing data to provide initial solution for new projects without historical data
This methodology prevents defects from happening rather than removing them after they have happened, Quality improvement, cost reduction	T13	The role of other factors such as team size or CMM level are needed to be discovered

4. Defect Prevention Flow

Defect prevention is a quality assurance method for improving the quality of the software product which makes it one of the most important activities in any software project [46]. Defect Prevention is identified as a level 5 Key Process Area (KPA) in the Capability Maturity Model (CMM) by the Software Engineering Institute, which involves analysing defects and take action to prevent the recurrence of similar type of defects in the future [49]. Analysis of defects at early stages reduces time, cost and resources and in the end enhances the overall productivity.

From the logical viewpoint, the flow of Defect Prevention can be elaborated as shown in Fig 3

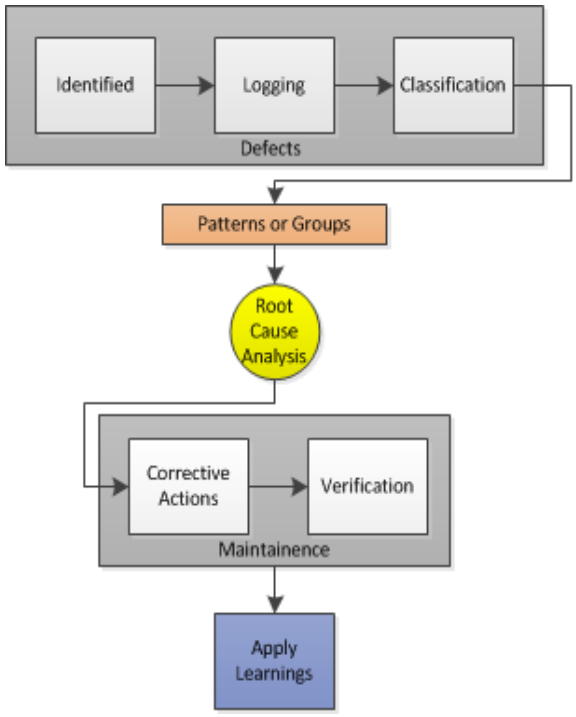

Figure 3: Defect Prevention Flow

The various steps of DP flow are elaborated below:

- Data from past projects, UAT reports, meetings and inspections are taken, and defects are identified.
- All the defects should be logged through the Defect logging system regularly and a considerable amount of information is recorded to facilitate tracking or resolution [18].
- Defects are classified using some good classification scheme like Orthogonal Defect Classification (ODC). ODC attributes extracted from the defects provide an enormous amount of information from individual defect [12] and defect type distribution i.e. defect signature would be generated. If adequate information is collected on each defect found and fixed, one can easily exploit ODC-based analysis in a very short time [37].
- With the use of ODC, it is possible to arrive at patterns and do Root Cause Analysis (RCA) on them. RCA plays an important role in finding the root cause of the problem and initiate action or corrective measures to eliminate the source of defect [7].
- Apply the learning's of the projects as precautionary ideas in similar projects [38].

Through the study of various papers, it has been observed that ODC in conjunction with RCA is quite advantageous, ODC can alter the economics and viability of root cause analysis by reducing the time it takes to perform the work and allow for greater coverage of the defect space [39]. RCA is a staple diet for the improvement of software development process and ODC helps reduce the cost of RCA while increasing its coverage [40].

Defect Prevention is one ignored part in some of the projects. In actual practice, DP teams should be identified, and a kick-off meeting should be held in starting to raise awareness and identify solutions, training on DP and causal analysis should be given [34]. DP teams should meet often to identify the problems and find the root cause of the problems. The aim is to improve software quality by using readily available data to decrease defects injected and increase defects detected which can be done by applying ODC to DP process [41]. An automated tool like Performance and Continuous Re-Commissioning Analysis Tool (PACRAT) is created for implementation of DP and defect detection activities [33].

5. Pillars of Defect Prevention – T3

Based on the learning's from a set of papers, three T's have been identified which are three important pillars for the improvement of the Defect Prevention approach as shown in Fig 4.

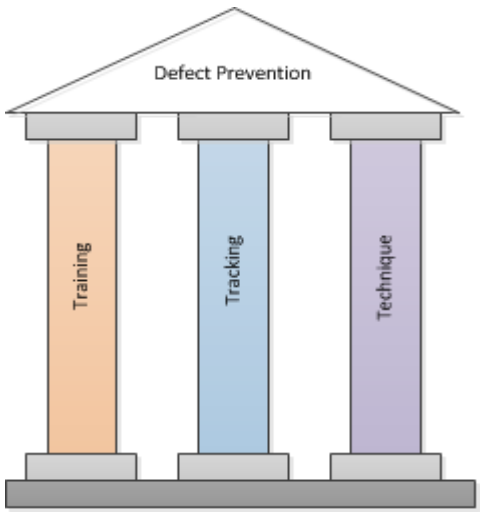

Figure 4: Pillars of Defect Prevention

Training is identified as one critical point that needs attention. A proper training is required to all the stakeholders so that they understand the use of proper defect tracking and technique of defect prioritization, classification and prevention. Defect Prevention is applied everywhere, but the manner of application is important. Hence adequate training should be given to everybody involved in projects like tester or developer or manager about how important Defect Prevention is, how it can help to reduce the rework when applied properly.

Tracking of all the defects is important as we want everybody to get complete and consistent information about them like the origin of the defect, phase of detection, priority, defect types etc. Defect tracking provides valuable information to improve software processes and streamline the process of reporting, handling and fixing issues. They should be chosen based on user's requirement and constraint [35]. There are various defect tracking system, however, more important is users must be motivated to make the correct use of defect tracking system by refilling the data after the defect is re-examined or updated.

Another important point is the use of the proper defect classification technique that can be applied on the defect. There are different types of defect classification techniques like IBM's Orthogonal Defect Classification or HP approach called Defect Origins, Types and Modes. Stefan Wagner and Huber have summarized these two classification techniques along with the comparison and current challenges in this aspect [10][42]. Defect classification helps in deciding which defect to correct firstly and which one can be ignored for later correction. There are lots of case studies which demonstrated the use of ODC for improving software quality [43]. The approach is to analyse defects by categorizing them and creating a distribution chart about the process However, it has also been found that general defect type classifications are difficult to use and need to be developed or modified for specific project domain and environment [45].

Some of the learnings of this study are also listed below:

1. Make developers, testers and managers realize the importance of Defect Prevention.
2. Use simplified but Elaborative Defect Tracking system.
3. A case study of defect detection and analysis techniques in previous projects should be given to stakeholders.
4. One good defect classification technique should be applied in projects.

6. Conclusion

It is very important to encourage defect preventive practices in various software projects so as to reduce defects for improving the quality of software by reducing the cost, time and rework. This paper gave a conceptual view of the defect management, defect prevention, classification and processes used. Through this study, work done by different authors have been summarized, and various techniques have been identified which are critically analysed in this paper. Through T3 approach, better quality in software projects can be achieved by focusing more on few aspects which are often ignored due to lack of time but when taken seriously can give commendable results.

A structured way of training on proper defect tracking, defect classification technique and root cause analysis

and their implementation in software projects of various organizations is suggested. The techniques used and the results obtained can be a boost for people to try ODC, RCA, inspection with other techniques discussed above in their own projects to set up a structured process. Therefore, we propose to invest more effort in tracking the defects and trying some standardized classification techniques. Last but not the least, imbibing the users to use processes as they are easy.

References

[1] Wheeler S and Duggins S. Improving software quality[C]. ACM Southeast Regional conference, 1998, 300-309.

[2] Li M, Xiaoyuan H and Sontakke A. Defect Prevention: A General Framework and its Applications[C]. Proceedings of the IEEE sixth International Conference on Quality Software, Beijing, China, Oct. 2006,281-286.

[3] Kumaresh S. and Ramachandran B. Defect Prevention based on 5 dimensions of Defect Origin[J]. International Journal of Software Engineering & Applications (IJSEA),2012,3(4).

[4] Mays R.G. Applications of Defect Prevention in Software Development[J]. IEEE J. Selected Areas in Communications,1990,8(2):164–168.

[5] Suma V and Gopalakrishnan Nair T.R. Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels[J]. Proceedings of World Academy of Science, Engineering and Technology,2008,32

[6] Kumaresh S. and Baskaran R. Defect Analysis and Prevention For Software Process Quality Improvement[J]. International Journal of Computer Applications,2010,8(7):42–47.

[7] Kalinowski M., Card D. and Travassos G.H. Evidence –Based Guidelines to Defect Causal Analysis[J]. Software, IEEE,2012,29(4):16-18, doi: 10.1109/MS.2012.72.

[8] Dalal S and Chhillar R.S. Empirical Study of Root Cause Analysis of Software Failure[C]. ACM SIGSOFT Software Engineering Notes archive, July 2013,38(4):1-7.

[9] Leszak, M., Perry D., and Stoll, D. A Case Study in Root Cause Defect Analysis[C]. Proceedings of the 22nd International Conference on Software Engineering, ACM Press, June 2000, 428-437.

[10] Stefan Wagner. Defect Classification and Defect Type Revisited[A]. Proceedings of the 2008 workshop on Defects in large software systems, (DEFECTS¨08), ACM Press, 2008.73-83.

[11] Chillarege Ram. Orthogonal Defect Classification[M]. In M. R. Lyu, editor, Handbook of Software Reliability Engineering, chapter 9. IEEE Computer Society Press and McGraw-Hill, 1996.

[12] Chillarege R., Bhandari I.S., Chaar J.K., Halliday M.J., Moebus D.S., Ray B.K. and Wong M.Y. Orthogonal Defect Classification-A Concept for In-Process Measurements[J]. IEEE Transactions on Software Engineering, 1992, 18(11):943-956.

[13] Shenvi A. Defect Prevention with Orthogonal Defect Classification[C]. In Proc- ISEC '09, Feb 2009,83-88.

[14] Trivedi P. and Pachori S. Modelling and Analysis of Software Defect Prevention using ODC[J]. International Journal of Advanced Computer Science and Applications, 2010, 1(3) :75-77

[15] Zhi-bo L, Xue-mei H, Lei Y, Zhu-ping D and Bing X. Analysis of software Process Effectiveness Based on Orthogonal Defect Classification[C]. 3rd International Conference on Environmental Science and Information application Technology(ESIAT 2011), Elsevier, 2011, 765-770.

[16] Bridge N and Miller C. Orthogonal Defect Classification Using Defect Data to Improve Software Development[C]. International conference on software quality, Oct 6-8, 1997,7(0),197-213.

[17] Biolchini J et al. Systematic Review in Software Engineering[R]. Tech. report ES 679/05-PESC/COPPE/UFRJ, Federal Univ. of Rio de Janeiro, 2005.

[18] Jalote P., Munshi R. and Proebsting T.A. The When-Who-How analysis of defects for improving the quality control process[J]. Journal of Systems and Software,2007,80(4):584-589.

[19] Li J, Stålhane T, Conradi R and Kristiansen J.M.W. Enhancing Defect Tracking Systems to Facilitate Software Quality Improvement[J]. IEEE Software, 2012, 29(2): 59-66.

[20] Mittal S., Solanki K and Saroha A. menment of Defects for Improving Software Processes[J]. International Journal of Computer Science and Management Studies, 2011,11(2):2231-5268.

[21] Khan H.A. Establishing a Defect Management Process Model for Software Quality Improvement[J]. International Journal of Future Computer and Communication, 2013,2(6):585-589.

[22] Card D.N. Managing software quality with defects[J]. The Journal of Defence Software Engineering,2003.

[23] Lehtinen, T., Mäntylä, M.V. and Vanhanen, J. Development and evaluation of a lightweight root cause analysis method (ARCA method) - Field studies at four software companies[J]. Journal of Information and Software Technology, 2011,53(10):1045-1061.

[24] Taba N.H. and Ow S.H. Software Defect Management Using a Comprehensive Software Inspection Model[J]. Software Engineering,2012, 2(4):160-164 DOI:10.5923/j.se.20120204.09.

[25] Langari, Z. and Pidduck, A. B. Quality, cleanroom and formal methods[C]. SIGSOFT Softw Eng. Notes, May 2005, 30(4) ,1-5. DOI= http://doi.acm.org/10.1145/1082983.1083302.

[26] Huang L, Ng V, Persing I, Geng R, Bai X, and Tian J. AutoODC:Automated Generation of

Orthogonal Defect Classifications[C]. Proceedings of the 26th IEEE/ACM Int. Conf. on Automated Software Engineering, Lawrence, KS, USA,2011,412-415.
[27] Dubey A. Towards adopting ODC in automation application development projects[C]. In Proceedings of the 5th India Software Engineering Conference, New York, USA, 2012,153–156
[28] Basin K and Santhanam P. Managing the Maintenance of Ported, Outsourced, and Legacy Software via Orthogonal Defect Classification[C]. In Proceedings of the IEEE International Conference on Software Maintenance, Washington DC, USA, 2001, 726. IEEE Computer Society,2001.
[29] Card D. Defect Causal Analysis Drives Down Error Rates[J]. IEEE Software,1993,10(4):98–99.
[30] Suma V and Nair T.R.G. Defect Management Strategies in Software Development[R]. CoRR abs/1209.5573, 2012.
[31] Potnuri D, and Stringfellow C.V. Analysis of Open Source Defect Tracking Tools for Use in Defect Estimation[C]. Software Engineering Research and Practice, CSREA Press, 2005, 296-301.
[32] Tiejun P, Leina Z and Chengbin F. Defect Tracing System Based on Orthogonal Defect Classification[C]. In Proc. International Conference on Computer Science and Software Engineering, 2008, 2, 574-577.
[33] Bean E. Defect Prevention and Detection in Software for Automated Test Equipment[J]. Instrumentation & Measurement Magazine, IEEE,2008,11(4):16-23.
[34] Jalote P and Agarwal N. Using Defect Analysis Feedback for Improving Quality and Productivity in Iterative Software Development[C]. In proc- ITI 3rd International Conference on Information and Communications Technology, Dec 2007, 703-713.
[35] Singh V.B. and Chaturvedi K.K. Bug Tracking and Reliability Assessment System (BTRAS)[J]. International Journal of Software Engineering and Its Applications, 2011,5(4):1-14.
[36] Chang C.P. And Chu C.P. Defect prevention in software processes: An action-based approach[J]. Journal of Systems and Software,2007, 80(4): 559-570.
[37] Bassin K.A, Kratschmer T, and Santhanam P. Evaluating Software Development Objectively[J]. IEEE Software,1998,15(6):66-74.
[38] Sharma A., Hemrajani N., Shiwani S and Dave R. Defect Prevention Technique in Test Case of Software Process for Quality Improvement[J]. International Journal of Computer Technology and Application,2012, 3(1):56-61.
[39] Ram Chillarege. ODC - a 10x for Root Cause Analysis[A]. Proceedings RAM 2006 Workshop, Berkeley CA, 2006.
[40] Chillarege R. ODC Measurement and Analysis - Industry Applications[R]. Technical report, Chillarege Inc., 2007.
[41] Graham M. Software Defect Prevention using Orthogonal Defect Prevention[R]. 2005, http://twin-spin.cs.umn.edu/node/844.
[42] Huber, J. T. A comparison of IBM's orthogonal defect classification to Hewlett Packard's defect origins, types and modes[C]. In Proceedings of International Conference on Applications of Software Measurement, San Jose, CA, 2000, 1-17.
[43] Buther M, Murino M. and Kratcher T. Improving Software Testing using ODC - Three Case Studies[J]. IBM Systems Journal,2002, 41(1):31-44.
[44] Leszak M., Perry D., and Stoll D. Classification and Evaluation of Defects in a Project Retrospective[J]. Journal of Systems and Software, Elsevier,2002,61(3):173-187.
[45] Freimut B, Denger C, and Ketterer M. An Industrial Case Study of Implementing and Validating Defect Classifcation for Process Improvement and Quality Management[C]. In Proc.11th IEEE International Software Metrics Symposium (METRICS '05), IEEE Computer Society, Sept 2005,19.
[46] Kavitha D. and Sheshasaayee A. Literature Review on Defect Management Process[J]. European Journal of scientific Research,2012, 85(3):426 – 431.
[47] http://www.chillarege.com/odc.
[48] http://researcher.watson.ibm.com/researcher/view_project.php?id=480.
[49] http://www.isixsigma.com/industries/software-it/software-defect-prevention-nutshell/.

Authors' Profiles

Chetna Gupta is Assistant Professor (Senior Grade) at Jaypee Institute of Information Technology, India. She obtained her Doctorate in the area of Software Testing. She also holds a Masters of Technology and a Bachelor of Engineering degree in Computer Science and Engineering. Her areas of interest are Software Engineering, Requirement Engineering, Software Testing, Software Project Management, Data Structures, Data Mining and Web Applications. She has many publications in international journals and conferences to her credit.

Priyanka Chandani is Assistant Professor at JSS Academy of Technical Education, India. She also holds a Masters of Technology and a Bachelor of Engineering degree in Information Technology. She has also worked at Infosys Technologies and TechMahindra. She is pursuing her Ph.D from Jaypee Institute of Information Technology.Her research interest includes Software Quality, Software Testing and Requirement Engineering

How to cite this paper: Priyanka Chandani, Chetna Gupta,"A Survey on Effective Defect Prevention - 3T Approach", IJIEEB, vol.6, no.1, pp.32-41, 2014. DOI: 10.5815/ijieeb.2014.01.04

