Software Defect Analysis: A Review

Dheeraj [1], Chhavi Rana*[2]
*Assistant Professor, Department of Computer Sciences and Engineering, M. D University, Rohtak
*University Institute of Engineering and Technology
Maharshi Dayanand University, Rohtak (Haryana), India

Abstract— The primary aim of the successful software development organizations is to provide reliable software. An extension in the field of software testing process has ensured better scope, however a few sections of a software framework are observed to be more helpless to defects than others, and discovery of these can significantly elevate the designers to convey exclusive expectation software. The software business spends more cash on finding and settling bugs than for some other known cost driver. This ought not be the situation. Different methods can be utilized to guarantee the nature of software. Defect Prevention is a crucial undertaking in any software venture. Defect Prevention includes examining defects that were experienced because of the inaccurate prerequisites, defects in outline and distinctive kinds of codes. Insufficient defect counteractive action and deficient pre-test defect expulsion are emphatically corresponded with failure to gauge defect evacuation effectiveness. Defect Detection procedure is utilized as a part of a considerable lot of the software ventures to recognize the defects, report such defects and break down them in order to refine the item and improve it. The defect archive will be a contribution for defect analysis. In Defect Analysis procedure, investigating the defect prompts distinguishing its underlying driver and to discover an answer for the defect along these lines keeping it from proliferating into encourage improvement stages. Arrangement of defects into classes and finding the conceivable reason are parts of defect analysis. Reckoning and counteractive action of defects before their event is the primary point of Defect Prediction strategy. In this paper, we have checked on Software Defect Prevention by its analysis and prediction and furthermore featured the models of defect prediction examines over the most recent fifty years. These strategies help in early recognition of defects, counteracting them to spread to next improvement stages and repressing the occurrence of comparative defects in future activities.

Keywords- defect, prediction model, detection, pre-and post-release, defect prevention

INTRODUCTION

SOFTWARE DEFECT

Software defect is an undesired element in a software which happens due to the occurrence of an error in the software development process. It might be anything from an error in the code to any incorrectly specified requirements [1]. A software defect is a flow, failure, error and faults in a program of a computer or system that results in an unexpected outcome or causes the program to perform in undesired ways. If a problem arises in the tested component then only that problem is called defect or bug [2]. An defect may begin in one development stage and be found in the same particular or a later stage. Techniques that can be used to detect a defect are inspection and testing. In testing, there are two types of testing, one is structural and another is functional testing. It is a discrepancy or imperfection which arises in a software work product (SWP). SWP is a set of methods and transformations taken up by people who build up and maintain the software [3]. To make the software deliverable as well as defect free, meticulous testing has to be carried out. This is a vital part of the software engineering process [4]. Requirement defects, Design defects and code defects are three types of software defects.

TYPES OF SOFTWARE DEFECT [5,6]

I. **Requirement defects:** These kinds of defects occur when incorrect requirements are established. The best way to detect such type of defects is by inspection. Testing can prove to be costly because developing a system on a bad set of requirements and then when it fails, having to re develop it.

II. **Design defects**: These defects happen when the framework is despicably outlined. Different observational examinations have affirmed that reviews were essentially more compelling and productive than testing. An imperfection recognized amid outline investigation is less expensive to adjust than one identified in work testing, on the grounds that the cost of modify in the last is fundamentally higher.

III. **Code defects:** For these type of defects, functional or structural testing is found to be better than inspection. Some studies claim that testing and inspection find different kinds of code defects and hence can be used together to complement each other.

SOFTWARE MODULES

A module is a part of a program that summarizes the data and code to execute a particular functionality and has an edge that allows the clients to uniformly access its functionality. It is easily pluggable with the other expecting interface modules and is simply enclosed in a single unit so that it can be deployed trouble-free. One or more than one routines can be included in a single module.

PRE-RELEASE SOFTWARE DEFECTS

The defects which are discovered by the software testing team before the release of the software to the clients are known as pre-release software defects. Similar terms used in research publications for pre-release software defects are in house defects or faults. The strength of the testing process can be evaluated by the number of pre-release software defects.

POST-RELEASE SOFTWARE DEFECTS

The defects obtained by customers after using the product are known as post-release defects [7]. These defects are also called as field defects because of their occurrence in the field found by clients. Weaknesses in the coverage of the performed test activities showed a large number of post-release deformities/defects.

PRE-RELEASE TESTING

Pre-release field testing generally called beta testing, is carried out near completion of the new version of the application. It validates the utilization of application in different real-life circumstances and under various configurations. It can last up to few weeks. Throughout this phase users without any professional background are being provided with instrumented versions of the application. Then the development team is presented with recorded problems noted by instrumental versions [8]. Hangs and crashes during the operation of application are among the major part of the reported problems. The primary key components of detailed issue are data which expresses the season of issue and the stack follow preview from the entry of the said issue.

FUNCTIONAL TESTING

It is one of the types of black box testing which stand its test cases on the requirements of the software component under test. It includes the identification of functions, generation of input data, determination of output, implementation of test case and finally comparing the actual and anticipated output.

ACCEPTANCE TESTING

After the completion of system testing acceptance testing is carried out. It generally establishes the reliability of the product delivered, by fulfilling the customer needs. The users and stakeholders are the one who performs the real acceptance testing. As the user stories passes their acceptance criteria, it is being depicted that the work done by the developers is progressing in the right direction.

SYSTEM TESTING

Complete integrated software testing is known as system testing. It also falls under black box testing. Based on SRS software requirements specification the behavior of fully integrated software product is tested. End user requirements is the major point in this testing

MODELS OF DEFECT PREDICTION

Software defect can be prevented by its analysis and prediction. Over the years different processes have been applied in software defect prediction. Various models of defect prediction studies in the last fifty years are presented in table 1.

Table 1Models of Defect prediction during last fifty years

S. No	Models of defect prediction	Studies conducted
1.	Line of code model	**In 1971, Mr. Akiyama** inferred that imperfections can be caused because of complex source code. He constructed an essential model by lines of code (LOC). It proposed the product frameworks intricacy. In spite of the fact that, LOC is excessively gullible metric, making it impossible to decide the many-sided quality of frameworks [8]
2.	Fitting model	**In1976 and 1977, MaCabe and Halstead** proposed two metrics named as cyclomatic many-sided quality metric and Halsted multifaceted nature measurements. The creators examined connection between the two measurements and the quantity of deformities. The creators did not try this model on new programming modules [9].
3.	Linear regression model	**In 1980, Shen et al** assembled a model named as direct relapse and furthermore tried that model on new programming modules to decide the issues that happened in the past models. [10].
4.	Classification model	**In 1981, Munson et al** proposed the order display which characterized the product parts into two noteworthy gatherings named as High Risk gathering and the Low Risk gathering. The arrangement shows created in this investigation acquired ninety-two percent of exactness which was very high around then. In any case, the investigation of these creators still had different hindrances as no measurements was utilized for object oriented framework modules. [11].
5.	Different Process model	**In 1990**, version control frameworks additionally were getting favored, and advancement stage in programming lifecycle was connected with the product repositories so unique process metrics can be utilized to decide the imperfections happening in software**.** [12].
6.	Object-oriented model	**In 1994**, several object-oriented metrics in terms of OO systems were put forward by **Chidamber and Kemerer** and **Basili et al.** took this model into consideration for predicting software defects in object oriented system [13].
7.	Artificial Neural Network model	**In 1997**, Prediction of software quality was presented by **Khoshgaftaar et al**. by utilizing simulated neural system. In this model, they characterized modules as fault level or non-fault level, utilizing large telecommunication software system. They compared their final products and another non-parametric model got from discriminant technique [14]

8.	Prediction model	**In 2000s**, **Zimmermann et al.** proposed new matrix, privacy issue of defect was engaged by Peters et al. There happened a great deal of restrictions for deformity expectation. With respect to the primary constraint which was that if prediction model can be utilized before the arrival of the product so that quality could be ensured. The second impediment was that for new undertaking or the improvements having less or no chronicled information the prediction model was difficult to build. As the act of process measurements was getting well known, In the investigations of software imperfections and defect prediction this confinement ends up being the greatest and the most troublesome issue [15].
9.	Just In Time (JIT) defect prediction model	**In 2000s**, to predict defects by changing the source code was found to be much more helpful. To change the source code, researchers projected a defect prediction model. Of late, the sort of models suggested are called as Just In Time (JIT) defect prediction models. Different scientists in the course of recent years have contemplated the JIT prediction models [16].
10.	Cross-project defect prediction models	Zimmermann et al. and He et al directed the learning on cross-prediction feasibility. Data researchers proposed various cross-project defect prediction models to form a prediction model for latest developments and the projects having no or less ancient data. Detecting cross-prediction remained another issue in cross- project defect prediction [16].
11.	Personalized defect prediction model and universal model	New ideas of prediction models have been offered, for example, customized defect prediction model and all-inclusive model
12.	Probabilistic Model	**In 2001**, For estimation of faulty software modules the usage of Bayesian belief networks (BBN) is suggested by **Fenton et al.** They gave this Probabilistic Model for Software Defect Prediction. They advanced that these models miss the mark to assess all the causal or illustrative factors required keeping in mind the end goal to develop the models generalizable [17].
13.	Defect detection model	**In 2002,** the importance of using defect detection techniques for each phase in a life cycle model was stated by **J.H. van Moll et al.** Thus, preventing most of the defects from propagating to the later stages [18].
14.	Usage-based reading mode	**In 2003,** Principles from programming assessment, use cases, and operational profile testing were consolidated into the use based reading strategy by Claes Wohlin et al. The objective was to give an effective reading technique to software examinations, which took the client perspective on the product and the shortcomings it may contain. The client reads, for instance, a design document guided by prioritized use cases. An exploratory assessment demonstrated that the UBR technique was more viable and effective in discovering flaws, critical to the client, contrasted with agenda based strategies [19].
15.	Evaluation model	**In 2003,** an experimental assessment of investigation and testing for location of configuration deficiencies was finished by H. Petersson et al. [20].
16.	RF model	**In 2004**, RF technique for forecast of defective modules with NASA statistics sets was employed by **G. John, P. Langley [**21].
17.	Software science ineffectiveness model	**In 2009**, unproductiveness usage of software science metrics as forecasters of defects in object oriented software was discussed by **Z. A. Rana, S. Shamail and M. M. Awais [22].**
18.	Defect analysis model	**In 2010,** a study on Defect investigation and prevention for software process quality enhancement was done by **S. Kumaresh et al.** [23].
19.	Orthogonal Defect Classification (ODC) model	**In 2011,** another approach for foreseeing the defects and arrangement of defects in light of undertaking qualities in the early life cycle stages was given by P.R. Anantula et al. Utilizing Orthogonal Defect Classification (ODC) approach, utilization of the current proposed strategies to enhance and understand the quality being developed and test conditions in a web application was clarified. A detail depiction was given on how the product procedure model adjusts itself to business objectives and accomplishes great quality programming by foreseeing the number and sort of defects well ahead of time important move made to lessen the event of defects. [24]

20.	Software quality improvement model	**In 2012**, causative factors which suggested the solutions to increase software quality and throughput were identified by **Rawat et al.** They clarified that how the different defect prediction models are executed bringing about decrease in size of defects [25].
21.	Defect Density Prediction model	**In 2013,** to focus on the defect density region, two primary methodologies powerful and functional were defined and connected to anticipate the presence of defect in document of programming discharge by V. Kumar et al., but the appropriateness of best approach relied upon the information accessibility and quantum of information [26].
22.	Preventive actions model	**In 2016,** a survey on the various techniques to prevent software defects through defect detection, analysis, and prediction was conducted by **Manjula R et.al.** [1]. Execution of defect preventive action not merely helped to give a quality project, but it was also an appreciated outlay. The preventive actions that were proposed by **Manjula R et. al.** was restricted to only limited types of defects under separate group. There may be many additional defects that would progress at each phase Irrespective of the project life cycle model used, it was recommended to validate and verify each phase of the developmental phase so as to prevent the defect from propagating into later stages which prevents extensive rework. Through appropriate techniques of detection of defects, apt choice of defect analysis method of the logged defects and defect prediction, which helped in identifying where the defects were likely to occur, ensured early identification and rectification of the defect, thereby preventing the future occurrence of the defect.

METRICS USED

Two principle approaches Average utilization Time (AVT) and Mean Time between failures (MTBF) of the application amid the testing time frame are utilized to assess the application quality. MTBF means the average measure of time which goes between arbitrary failures of running application [27]. Along these lines this approach is utilized to foretell the quantity of post-release defects. AVT is the average time that an application is in effect effectively utilized by a client. It can last longer than field testing period, as the utilization of the application proceeds even after the field testing. It portrays that more extended the AVT more dependable the application is. Different measurements utilized for post-release defect prediction is Average Time to First Failure (TTFF) which measures the average measure of time before the client experience first failure. The all the more early the first failure

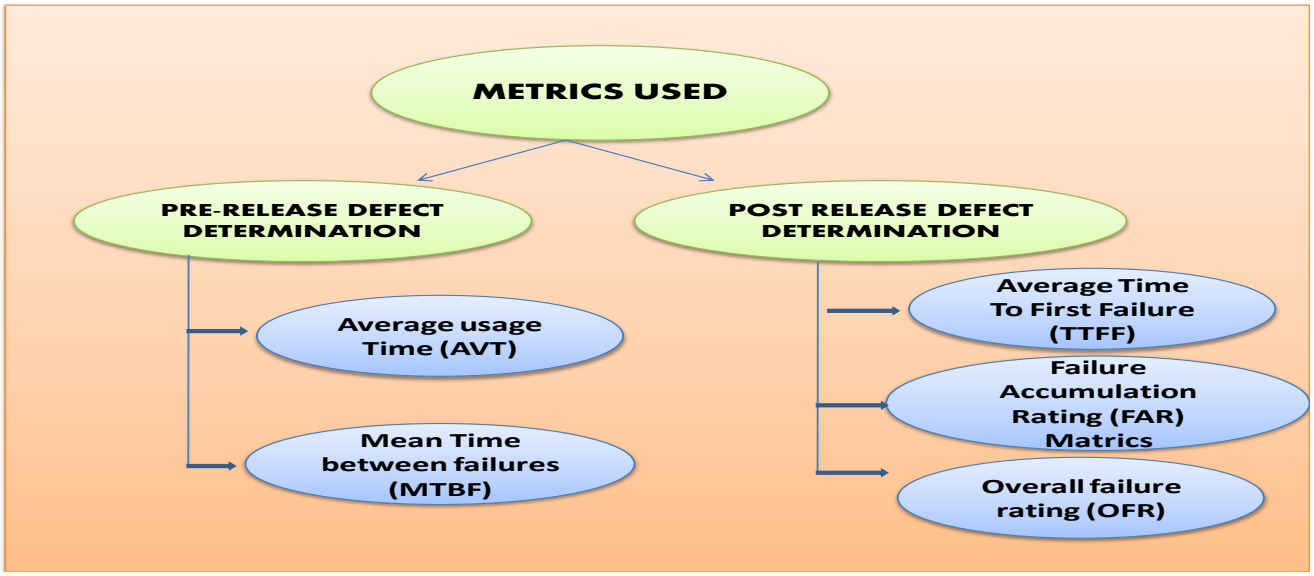

Fig. Metrics Used for Defect Determination of Defect

comes more buyers are worried about the dependability and nature of the application. [28] The failure accumulation rating (FAR) metric investigates the circulation of failures. It is utilized to discover the quantity of clients fall as far as total number of failures. Overall failure rating (OFR) limits the portion of the level of clients that report failures frequently amid the time of testing [29]. The OFR and FAR measurements are more critical to ascertain the quantity of post release defects than regular MTBF and AVT measurements [30].

CONLUSION

This paper summarizes the research findings of numerous researchers about software defect analysis. It gives vision about various software defects and there types along with different prediction models. The motive of this paper is to make user understand the concept about how software defects can be prevented by its analysis and prediction. The methods defined above helps in early exposure of defects, averting them to spread to the next progression stage and inhibiting the happening of related defects in forthcoming developments so that the time and efforts required for the maintenance of the software can be decreased without affecting the quality and reliability of the product.

Software defect prediction models have been considerably discussed in software testing domain and its future scope lies in combining the various pre-and post-defect analysis models and thus developing a new model thereby exploiting the advantages of the combined approach which will be beneficial for the software development industry.

REFERENCES

[1] R. Manjula, S. Karthik, A. Shenny and T.V. Kamal, "Software Defect Prevention through Defect Detection, Analysis, and Prediction: A Survey," vol. 5, 3rd Int. Conf. on Innovations in Information Embedded and Communication Systems (ICIIECS), Coimbatore, pp. 1-11, 2016.

[2] K. Sakthi, R. Bhaskaran, "Defect Analysis And prevention for software quality improvement," Int. J. Comp. App., vol. 8 (7), 2010.

[3] J. H. van Moll, J. C. Jacobs Philips, B. Freimut Fraunhofer, J. J. M. Trienekens, "The Importance of Life Cycle Modeling to Defect Detection and Prevention", Software Technology and Engineering Practice, 2002. STEP 2002. Proceedings.

[4] W. Humphrey, "The software quality challenge," Crosstalk; The journal of Defense Software Engineering, vol. 21, pp. 4-9, 2008.

[5] T. Berling and T. Thelin, "An Industrial Case Study of the Verification and Validation Activities," Proc. 9th Int'Software Metrics Symp., IEEE CSPress, 2003, pp. 226–238.

[6] C. Andersson et al., "An Experimental Evaluation of Inspection and Testing for Detection of Design Faults," Proc. IEEE/ACM Int'l Symp. Empirical Software Eng., IEEE CS Press, 2003, pp. 174–184.

[7] B. M. Parareda and M. Pizka, "Measuring Productivity Using the Infamous Lines of Code Metric, ‖ Software Productivity Analysis and Cost Estimation," 2007.

[8] S. Keene, "Modeling software R&M characteristics," ASQC Reliability Review, Part I and II, vol. 17, no. 2&3, pp. 13–22, June 1997.

[9] M. Shepperd, "A critique of cyclomatic complexity as a software metric," Software Engineering Journal, vol. 3 (7), pp. 30-36, 1988.

[10] V.Y Shen, T. J. Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying Error-Prone Software — An Empirical Study" IEEE Transactions on Software Engineering SE-11, vol. 4, pp. 317–323, 1985.

[11] J. C. Munson and T. M. Khosgoftaar, "The detection of fault prone programs," IEEE Transactions on Software Engineering, vol. 18, pp. 423–434, May 1992.

[12] K. S. Seija, "Development and Evaluation of Software Process Improvement methods," ESPOO, VTT publication, pp. 4-25, April 2004.

[13] S. Chidamber and C.F. Kemerer, "A metrics Suite for Object-Oriented Design", IEEE Trans. Software Engineering, vol. 20 (6), pp. 476-493, 1994.

[14] T.M. Khoshgaftaar, E.D. Allen, J.P Hudepohl, S.J Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.

[15] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In Proceedings of the 30th international conference on Software engineering, ICSE '08, pp. 531–540, 2008.

[16] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large-scale experiment on data vs. domain vs. process. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ESEC/FSE '09, pp. 91–100, New York, NY, USA, ACM, 2009.

[17] Norman Fenton, Paul Krause and Martin Neil, "A Probabilistic Model for Software Defect Prediction", IEEE Transactions on Software Engineering, 2001.

[18] J.H. Van Moll, J.H. Jacobs, B. Freimut and J.J.M. TrieneKens, "The Importance of Life Cycle Modeling to Defect Detection and Prevention," Proceedings of the 10 International Workshop on Software Technology and Engineering Practice (STEP'02), IEEE Computer Society 2002.

[19] C. Wohlin, T. Thelin, P. Runeson, T. Olsson and C. Andersson, "Evaluation of Usage-Based Reading—Conclusions after Three Experiments "Empirical Software Engineering, vol. 9, pp.77-110, 2004.

[20] H. Petersson, T. Thelin, P. Runeson, and C. Wohlin, 2004. "Capture-recapture in software inspections after 10 years research-theory, evaluation and application," Journal of Systems and Software, 2004.

[21] L. Guo, Y. Ma, B. Cukic, H. Singh, "Robust prediction of fault proneness by random forests," In: Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04), pp. 417–428, 2004.

[22] Z. A. Rana, S. Shamail and M. M. Awais, "Ineffectiveness of Use of Software Science Metrics as Predictors of Defects in Object Oriented Software," Proceedings of the 2009 WRI World Congress on Software Engineering, Vol. 04, pp. 3-7, 2009.

[23] Kumaresh, S. and R. Baskaran, 2010. Defect Analysis and Prevention for Software Process Quality Improvement. International Journal of Computer Applications, 0975-8887, 8(7): 42-47.

[24] Pranayanath Reddy Anantula, Raghuram Chamarthi. "Defect Prediction and Analysis Using ODC Approach in a Web Application" International Journal of Computer Science and Information Technologies, vol. 2 (5), pp. 2242 –2245, 2011.

[25] Mrinal Singh Rawat, Sanjay Kumar Dubey "Software Defect Prediction Models for Quality Improvement: A literature Study." IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012

[26] V Kumar, A. Sharma and R. Kumar, "Applying Soft Computing Approaches to Predict Defect Density in Software Product Releases: An Empirical Study" Computing and Informatics, Vol. 32, pp. 203–224, 2013.

[27] J. Musa, A. Iannino and, K. Okumoto, "Software Reliability. Measurement, Prediction, Application," McGraw-Hill, 1987

[28] A. Mockus, and D. Weiss, "Interval quality; relating customer perceived quality to process quality," in ICSE 2008; Proceedings of the 30th International conference on software engineering. NY, USA; ACM, pp. 723-732, 2008.

[29] Foutse Khomh, "Predicting Post-Release Defects Using Pre-Release Field Testing Results," ICSM, 2011.

[30] Phil Bantz, "Management and software quality assurance; A White Paper," Software Quality Assurance Subcommittee of The Nuclear Weapons Quality Managers. Quality Reports, SQAS92-002, June 1993.