Classification and Advantage of Software Fault Prediction

[1]Dr Mukesh Singla, [2]Parul

[1]Professor & Dean, [2]Research Scholar
[1]Faculty of Engineering, Baba Mast Nath University
Department of Computer Science and Engineering, Rohtak, India

***Abstract*: Software testing is tedious, resource consuming but important process in the software development lifecycle. Ideally, a large portion of software development resources is allotted to the software testing and software quality assurances activities [1]. Modern day software systems are growing complex and larger in size, thus making the task of software testing and quality assurance more difficult. The ultimate goal of the software testing process is to deliver the software system that is free from any bug and meeting the requirements of the stakeholders. However, ensuring that a software system is free from all faults is required an exhaustive and thorough testing, which is an expensive, almost impossible goal to achieve [2]. In real-world software projects, typically, a small amount of budget is left for the testing.**

***Index Terms*: Software testing, software systems, software projects etc.**

I. INTRODUCTION

The working of software fault prediction is founded on the assumption that if a previously developed software component was found faulty under certain environmental conditions, then any component currently under development with similar environmental conditions and with similar structural properties will end to be fault prone. Engineering is a problem-solving activity. Engineers search for an appropriate solution, often by trial and error, evaluating alternatives empirically, with limited resources and incomplete knowledge. During object design, developers define solution domain objects to bridge the gap between the analysis model and the hardware/software platform defined during system design. This includes precisely describing object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals such as extensibility or understandability, and optimizing the object model for performance. This includes precisely describing object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals such as extensibility or understandability, and optimizing the object model for performance.

II. CLASSIFICATION OF SOFTWARE FAULTS

In the software development process, a software fault refers to the mismatch between expected and actual output. It is also known as bug or defect. According to IEEE standards, a fault is "an incorrect step, instruction or data in a program" [3]. These faults eventually lead software system to the failure or sometimes may change the external behavior of the program. Black box testing detects functional-level faults and white box testing detects code-level faults. We include both types of faults in the software fault dataset. There many classifications of software faults are available based on their occurrence in the software development life cycle. Knowing the type of fault predicted by the model can leverage the tester in the selection of appropriate measures to locate and fix it. In this subsection, we discuss some of the classifications of faults available in the literature.

Orthogonal defect/fault classification (ODC) scheme has classified software defects into eight different categories [4]. The classification scheme is based on the semantics of defects. It means that a type of defects points to the part of the process that needs attention. The following classes of defects were proposed in this classification. Ning et al. have provided a complete list of ODC defects [5].

1. Function: It refers to the defect that affects end-user functionality such as an interface with hardware, interface with application program, etc., significantly.
2. Assignment: It refers to the defect that occurs in the few lines of code such as initialization of code or data structure.
3. Interface: It refers to the defect that occurs in the interface, which interacts with other components such as modules, device driver, etc.
4. Checking: It refers to the defect that occurs in the program logic, which fails to validate data and values, loop initial, and termination conditions, etc.
5. Timing/serialization: It refers to the defects occurs in the shared resources and global values.
6. Build/package/merge: It refers to the defect occurs due to the errors in library functions and packages.
7. Documentation: It refers to the defect occurs in the documentation and maintenance files.
8. Algorithm: It refers to the defects occurs due to the problem in the algorithm correctness and efficiency.

There are many techniques for increasing the reliability of a software system:

Fault avoidance techniques try to detect faults statically, that is, without relying on the execution of any of the system models, in particular the code model. Fault avoidance tries to prevent the insertion of faults into the system before it is released. Fault avoidance includes development methodologies, configuration management, and verification.Fault detection techniques, such as debugging and testing, are uncontrolled and controlled experiments, respectively, used during the development process to identify erroneous states

and find the underlying faults before releasing the system. Fault detection techniques assist in finding faults in systems, but do not try to recover from the failures caused by them. In general, fault detection techniques are applied during development, but in some cases they are also used after the release of the system. The blackboxes in an airplane to log the last few minutes of a flight is an example of a fault detection technique. Fault tolerance techniques assume that a system can be released with faults and that system failures can be dealt with by recovering from them at runtime. For example, modular redundant systems assign more than one component with the same task, then compare the results from the redundant components. The space shuttle has five onboard computers running two different pieces of software to accomplish the same task

III ADVANTAGES OF SOFTWARE FAULT PREDICTION

Software fault prediction is the key in improving the quality of the software system while optimizing the resource utilization and reducing the testing cost. Results of software fault prediction model allow the managers to make the decision about the allocation of development resources effectively and efficiently. Following are some advantages of software fault prediction in the development of software systems:

1. Reduction in testing efforts: Software testing consumes the biggest chunk of software development resources. In real-world projects, enough resources are not available to allot to the software testing. Software fault prediction (SFP) plays a crucial role in this situation. Early studies showed that SFP helps in reduction of testing efforts by a great amount. According to the study reported by Monden et al. [6], SFP can reduce total testing cost by 25%. However, authors reported that about 6% of the additional cost inures to the organization in terms of the collection of software metrics, preprocessing dataset, and building fault prediction model when using SFP along with software testing. Therefore, a total of 19% of testing efforts could be saved. In another study, Hryszko and Madeyski [7] showed that quality assurance cost has reduced by 30% when software defect prediction is used along with software testing. Additionally, the authors showed that the presented tool DePress has a return on investment of 73% and the benefit–cost ration of 74%. Moreover, authors deployed the tool to continue usage to the industrial projects run by Volvo group.

2. Early detection of faults and optimal allocation of resources: The primary aim of software fault prediction is to help the developer or software tester to identify risky areas of the software system to streamline their testing efforts. Boehm and Papaccio [8] found that removal of faults at design level is cost 50–200 times less costly than removal of faults after the deployment. A study conducted by NASA showed that a fault introduced in the requirement phase, if not removed and leaked into the later phases results in the much higher correction cost. The earlier studies showed that modules ranked as fault-prone by the software fault prediction model used by testers to detect additional faults in the modules that were considered previously to be a low fault-prone. Additionally, modules identified by testers as the most fault-prone are among the top four fault-prone modules identified by the software prediction model. In one of the study, Boehm's highlighted the importance of software fault prediction by showing that the verification effort saved by fault prediction model of correct identification of one fault is higher than the cost of misclassification of a hundred fault-free modules as fault-prone.

3. Improvement in the reliability of software systems: Nowadays, software is the key element of function in many modern engineered systems. In this growing number of applications of software, one vital challenge is to ensure the safety or the reliability of the software, i.e., the probability of failure-free operation for the specified period of time in the given environmental conditions. The reliability of the software depends on the software faults. More the faults, lessen the reliability of the software [9]. Many modern-day software engineering practices test the software under specified environment to ensure that software is correct or faultfree. However, it is an expensive and time-consuming process to test the whole system thoroughly due to the increasing size and complexity of the software. One of the potential solutions to this problem is to bring together two reasonings.

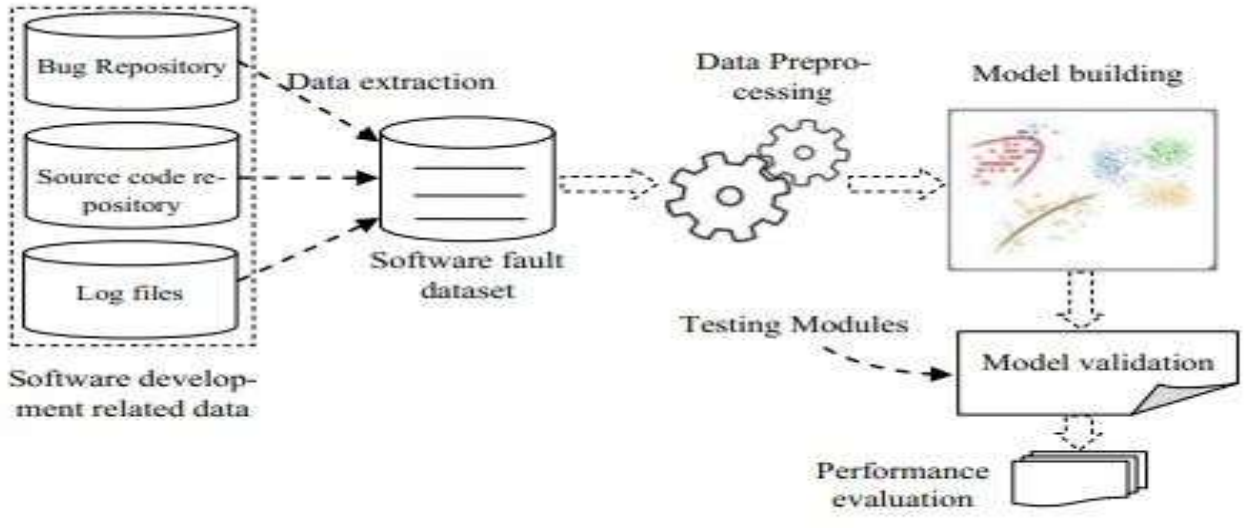

Fig. 1.1 Software fault prediction process

IV TECHNIQUES USED FOR THE PREDICTION OF NUMBER OF FAULTS

Prediction of number of faults refers to the process of estimating/predicting a potential number of faults that can occur in each given software module [10]. A software module can be a class for object-oriented software, file for traditional software or any other independent component having a bunch of code bundles together. This type of prediction can be very helpful for testers to focus on the software modules having more faults. One of the most popular methods for building this type of fault prediction models is the regression. Researchers for the software fault prediction have explored a large number of regression techniques and their variations. This includes techniques such as linear regression, ridge regression, generalized linear regression, Poisson regression, negative binomial regression, etc. [11]..Object-oriented design has become a dominant method in software industry and many design metrics of object oriented programs have been proposed for quality prediction, but there is no well-accepted statement on how significant those metrics are. The object oriented metrics will be adopted to identify a limited set of measureable attributes that have a significant impact on prediction of Faults and quality attributes. The techniques involved will be statistical analysis. The statistical techniques will be used to reveal the relationship between metrics and dependent variables.

V PERFORMANCE EVALUATION MEASURES

Once software fault prediction model is built, the next task is to assess its performance for some unseen examples (testing dataset) for unbiased evaluation. Here, fault prediction modeling follows a supervised model building process, where the dependent variable (number of faults) information is already available. We can use this ground truth information to compare the predicted values of faults with the actual value of faults and assess the performance of the prediction model. Performance evaluation measures are used for this task. Initially, original fault dataset is partitioned into the training dataset and testing dataset and then a fault prediction model is trained for the training dataset. Various performance evaluation measures are available that can be used to assess the performance of fault prediction model. In our fault prediction model, the dependent variable is of continuous type (regression), therefore, we have used error based performance evaluation measures. The used performance measures compared the predicted values of number of faults with the corresponding actual values of number of faults [12].

VI CONCLUSION

In the past few decades, software fault prediction has been one of the most active areas of software engineering research. It makes use of software structural- and product related metrics augmented with fault information to infer useful patterns and rules to predict fault occurrences or risky code areas in future software releases. This quality of software fault prediction can help developers to build software components having a lesser number of faults effectively and help managers to meet quality goals of under developing software in a cost-effective manner.

REFERENCES

[1] Adrion, W.R., Branstad, M.A., Cherniavsky, J.C.: Validation, verification, and testing of computer software. ACM Comput. Surv. 14(2), 159–192 (1982)
[2] Jiang, Y., Cukic, B., Ma, Y.: Techniques for evaluating fault prediction models. Empir. Softw. Eng. 13(5), 561–595 (2008)
[3] Huizinga, D., Kolawa, A.: Automated Defect Prevention: Best Practices in Software Management. Wiley (2007).
[4] Bridge, N., Miller, C.: Orthogonal defect classification using defect data to improve software development. Softw. Qual. 3(1), 1–8 (1998).
[5] Li, N., Li, Z., Sun, X.: Classification of software defect detected by black-box testing: an empirical study. In: Proceedings of IEEE Second World Congress on Software Engineering (WCSE), vol. 2, pp. 234–240 (2010)
[6] Monden, A., Hayashi, T., Shinoda, S., Shirai, K., Yoshida, J., Barker, M., et al.: Assessing the cost effectiveness of fault prediction in acceptance testing. IEEE Trans. Softw. Eng. 39(10), 1345–1357 (2013)Camargo Cruz Ana Erika, " Chidamber & Kemrer Suite of Metrics", Japan Advanced Institute of Science and Technology School of Information, May 2008.
[7] Hryszko, J., Madeyski, L.: Cost Effectiveness of software defect prediction in an industrial project. Found. Comput. Decis. Sci. 43(1), 7–35 (2018)
[8] Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IEEE Trans. Softw. Eng. 14(10), 1462–1477 (1988)
[9] Li, P. L., Herbsleb, J., Shaw, M., Robinson, B.: Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc. In: Proceedings of the 28th International Conference on Software Engineering, pp. 413–422 (2006)
[10] Ostrand, T.J., Weyuker, E.J., Bell, R.M.: Predicting the location and number of faults in large software systems. IEEE Trans. Softw. Eng. 31(4), 340–355 (2005)
[11] Rathore S.S., & Kumar, S.: An empirical study of some software fault prediction techniques for the number of faults prediction. Soft Comput. 21(24), 7417–7434 (2017a)
[12] Jiang, Y., Cukic, B., Ma, Y.: Techniques for evaluating fault prediction models. Empir. Softw. Eng. 13(5), 561–595 (2008)