International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 5 ǁ 2022 ǁ PP. 21-27

Study of improving software testability by Data Mining Techniques

Amitava Bondyopadhyay

Abstract
Software testing have a very important role in the process of building a high quality software. It is necessary because programmers, being human , commit mistakes. Sometimes those mistakes have some long lasting effect in the quality of the software. Hence in this paper a survey is made to see the significance of data mining techniques for software testing .
Keywords: *Cost, Classification, Data mining, Database, Dataset, Testing,Data Mining, Software Testing, Software Engineering, Test Case, Software Quality .*

Date of Submission: 18-04-2022 Date of acceptance: 03-05-2022

I. INTRODUCTION

Automated software testing can increase the depth and scope of tests to help improve software quality. Lengthy tests that are often avoided during manual testing can be run unattended. They can even be run on multiple computers with different configurations.

The main purpose of software development process is to develop high-quality software efficiently. In recent years the demand for software quality has increased quickly. In recent decades the production of large software projects are challenging , costly and time consuming. In order to minimize cost and enhancing the overall efficiency of the testing process , measuring software defects at early stage is particularly significant. So if we estimate at priori the probable faultiness of software , then it could end up giving help on software development activities planning, controlling and executing. Hence our aim should be to discover a low cost method that can be achieved from learning from earlier error to prevent future one. Nowadays quite a few data sets exists which could be mined to find useful facts about defects. From the beginning data mining techniques are applied in constructing software fault prediction for improving the software quality. So we need to identify high risk modules (having high number of faults) at the earliest which can be helpful in quality enhancement effort.

A software defect [16] is an fault, flaw, bug, mistake, breakdown, or fault in a computer program or system that may make an erroneous outcome, or precludes the software from behaving as wished **.**Any software development team for all time want to produce a quality software with least amount of defects. To boost the software quality, high risk modules from the software development should be take out at the earliest. Software defects constantly invite cost in terms of time and quality **.** To identify and correct defects is one of the major research areas nowadays. It is not possible to eliminate each and every defect in one software but it can be minimized and their adverse effects can be reduced.

Software engineering plays with the design and creation of programs for computers and other electronic devices. In a classic software development style, the work is split into distinct stages with exact activities in each, with the objective of improving planning and management. Such contrasting development paradigms and the intricate dependencies that they create increase the complexity of software systems. This slows down development and maintenance activities, causes faults and defects and eventually leads to an increase in cost of the software. Organizations often fail to understand how their process impacts the quality of the software that they produce. This is mainly due to the difficulty innate in discovery and measurement. Although software metrics have long been the de-facto standard for the assessment of software quality and development processes, their drawbacks are numerous. The over- reliance on metrics that can be easily obtained and understood, usage of metrics that seem interesting but remain irrelevant and uninformative and the difficulty in obtaining truly valuable metrics are but to name a few.

Data mining is defined as the process of discovering previously unknown and potentially useful information from data collections. Thus utilizing data mining in software testing with the aim of software improvement has piqued the interest of researchers worldwide. There are several challenges that emerge in mining software repositories. The major ones being, dealing with the inherent complexity and sheer volume of the software engineering data. Data mining concentrates on working with large quantities of data to provide a pattern. In terms of consumer data it is very useful to lead to attain successful marketing. So sometimes it is

violating the information protection law by proving unknown relationships in data[9]. For example, the software for pervasive computing based; HVAC control system to control energy consumption needs proper testing, to provide efficient energy consumption [3]. In this survey, we present an overview of data mining techniques and how they can be applied in the context of software testing. More specifically, we categorize these techniques with respect to the software development stages that they assist in the most.

II. STUDY OF IMPROVING SOFTWARE TESTABILITY

Test cases do not have the same significance when used to sense faults in software; therefore, it is more capable to test the system with the test cases that have the skill to detect the faults. This research proposes a new framework that combines data mining techniques to prioritize the test cases. It increases fault prediction and detection using two different techniques, firstly the data mining regression classifier[1] that depends on software metrics to predict defective modules, and the second one the k-means clustering technique that is used to choose and prioritize test cases to identify the fault at the beginning. Their approach of test case prioritization yields good results in comparison with other studies. The authors used the Average Percentage of Faults Detection APFD metric to evaluate the proposed framework, which results in 19.9% for all system modules and 25.7% for defective ones. The result gives an indication that it is useful to start the testing process with the most defective modules instead of testing all modules arbitrary .

The high reliability software is not only one of software technique development commanding points, but also is the software industry development essential foundation, [2] summarizes the data mining to face the detect of the software credibility test, the appraisal and the technical aspect newest research, elaborated the data mining technology in the software flaw test application, including flaw test in commonly used data mining method, data mining system and software testing management system. Introduced specifically in view of the software flaw's different classification based on the connection rule's software flaw parsing technique's application, proposed based on the association rule's software detect evaluation method, the purpose of which is to decrease software defects and to achieve the rapid growth of software dependability.

Software testing activities are usually planned by human experts, while test automation tools are limited to execution of pre-planned tests only. Evaluation of test outcomes is also associated with a considerable effort by software testers who may have imperfect knowledge of the requirements specification. Not surprisingly, this manual approach to software testing results in heavy losses to the world's economy. As demonstrated in this chapter, Data Mining algorithms(4) can be efficiently used for automated modeling of tested systems. Induced Data Mining models can be utilized for recovering system requirements, identifying equivalence classes in system inputs, designing a minimal set of regression tests, and evaluating the correctness of software outputs.

The primary goal of software development is to deliver high-quality software efficiently and in the least amount of time whenever possible. To achieve the preceding goal, developers often want to reuse existing frameworks or libraries instead of developing similar code artifacts from scratch. The challenging aspect for developers in reusing the existing frameworks or libraries is to understand the usage patterns and ordering rules among Application Programming Interfaces exposed by those frameworks or libraries, because many of the existing frameworks or libraries are not well documented. Incorrect usage of Application Programming Interfaces may lead to violated Application Programming Interfaces specifications, leading to security and robustness defects in the software. Furthermore, usage patterns and specifications might change with library refactoring, requiring changes in the software that reuse the library.

Data mining techniques are applied in building software fault prediction models for improving the software quality. Early identification of high-risk modules can assist in quality enhancement efforts to modules that are likely to have a high number of faults. The [5] presents the data mining algorithms and techniques most commonly used to produce patterns and extract interesting information from software engineering data. The techniques are organized in seven sections: classification trees, association discovery, clustering, artificial neural networks, optimized set reduction, Bayesian belief networks, and visual data mining can be used to achieve high software reliability.

To achieve the goal of creating products for a specific market segment, implementation of Software Product Line is required to fulfill specific needs of customers by managing a set of common features and exploiting the variability's between the products. Testing product-by- product is not feasible in Software Product Line due to the combinatorial explosion of product number, thus, Test Case Prioritization is needed to select a few test cases which could yield high number of faults. Among the most promising TCP techniques is similarity- based TCP technique which consists of similarity distance measure and prioritization algorithm. The goal of [6] is to propose an enhanced string distance and prioritization algorithm which could reorder the test cases resulting to higher rate of fault detection. Comparative study has been done between different string distance measures and prioritization algorithms to select the best techniques for similarity-based test case prioritization. Identified enhancements have been implemented to both techniques for a better adoption of

prioritizing Software Product Line test cases. Experiment has been done in order to identify the effectiveness of enhancements done for combination of both techniques. Result shows the effectiveness of the combination where it achieved highest average fault detection rate, attained fastest execution time for highest number of test cases and accomplished 41.25% average rate of fault detection. The result proves that the combination of both techniques improve Software Product Line testing effectiveness compared to other existing techniques.

In today's software industry, the design of test cases is mostly based on the human expertise, while test automation tools are limited to execution of pre-planned tests only. Evaluation of test outcomes is also associated with a considerable effort by human testers who often have imperfect knowledge of the requirements specification. Not surprisingly, this manual approach to software testing results in heavy losses to the world's economy. In [7] authors demonstrated the potential use of data mining algorithms for automated modeling of tested systems. The data mining models can be utilized for recovering system requirements, designing a minimal set of regression tests, and evaluating the correctness of software outputs. To study the feasibility of the proposed approach, they have applied a state-of-the-art data mining algorithm called Info-Fuzzy Network to execution data of a complex mathematical package. The Info-Fuzzy Network method has shown a clear capability to identify faults in the tested program.

In today's software industry, the design of test cases is mostly based on human expertise, while test automation tools are limited to execution of pre-planned tests only. Evaluation of test outcomes is also associated with a considerable effort by human testers who often have imperfect knowledge of the requirements specification. Not surprisingly, this manual approach to software testing results in heavy losses to the world's economy. In [8], they demonstrated the potential use of data mining algorithms for automated modeling of tested systems. The data mining models can be utilized for recovering system requirements, designing a minimal set of regression tests, and evaluating the correctness of software outputs. To study the feasibility of the proposed approach, they have applied a state-of-the-art data mining algorithm called Info-Fuzzy Network to execution data of a complex mathematical package. The Info-Fuzzy Network method has shown a clear capability to identify faults in the tested program.

Software engineering activities comprise of several activities to ensure that the quality product will be achieved at the end. Some of these activities are software testing, inspection, formal verification and software defect prediction. Many researchers have been developed several models for defect prediction. These models are based on machine learning techniques and statistical analysis techniques. The main objectives of these models are to identify the defects before the delivery of the software to the end user. This prediction helps project managers to effectively utilize the resources for better quality assurance. Sometimes, a single defect can cause the entire system failure and most of the time they drop the quality of the software system drastically. Early identification of defects can also help to make a better process plan which can handle the defects effectively and increase the customer satisfaction level. But the accurate prediction of defects in software is not an easy task because this is an indirect measure. Therefore, it is important to find suitable and significant measures which are most relevant for finding the defects in the software system. [10] Presents a feature selection based model to predict the defects in a given software module. The most relevant features are extracted from all features with the help of seven feature selection techniques and eight classifiers are used to classify the modules. Six NASA software engineering defects prediction data sets are used in their work. Several performance parameters are also calculated for measuring the performance and validation of this work and the results of the experiments revealed that their proposed model has more capability to predict the software defects.

Software engineering data contains a wealth of information about a project's status, progress, and evolution. Using well established data mining techniques, practitioners and researchers can explore the potential of this valuable data in order to better manage their projects and to produce higher quality software systems that are delivered on time and within budget. [11] Presents the latest research in mining Software Engineering data, discusses challenges associated with mining Software Engineering data, highlights Software Engineering data mining success stories, and outlines future research directions. Participants will acquire knowledge and skills needed to perform research or conduct practice in the field and to integrate data mining techniques in their own research or practice.

In today's industry, the design of software tests is mostly based on the testers' expertise, while test automation tools are limited to execution of pre-planned tests only. Evaluation of test outputs is also associated with a considerable effort by human testers who often have imperfect knowledge of the requirements specification. Not surprisingly approach to software testing results in heavy losses to the world's economy. The costs of the so-called catastrophic software failures are even hard to measure. Some of the researchers demonstrated the potential use of data mining algorithms for automated induction of functional requirements from execution data. The induced data mining models of tested software can be utilized for recovering missing and incomplete specifications, designing a minimal set of regression tests, and evaluating the correctness of software outputs when testing new, potentially flawed releases of the system. To study the feasibility of the proposed approach, they have applied a novel data mining algorithm called Info-Fuzzy Network to execution

data of a general- purpose code for solving partial differential equations. After being trained on a relatively small number of randomly generated input-output examples, the model constructed by the Info-Fuzzy Network algorithm has shown a clear capability to discriminate between correct and faulty versions of the program.

Satisfying the customer requirements is the ultimate goal of producing or developing any product. The quality of the product is decided on the bases of the level of customer satisfaction. There are different techniques which have been reported during the survey which enhance the quality of the product through software defect prediction and by locating the missing software requirements. Some mining techniques were proposed to assess the individual performance indicators in collaborative environment to reduce errors at individual level. The basic intention is to produce a product with zero or few defects thereby producing a best product quality wise. We can make analysis of survey the techniques like Genetic algorithm, artificial neural network, classification and clustering techniques and decision tree .. After analysis it has been found that these techniques contributed much to the improvement and enhancement of the quality of the product.

The increased availability of data created as part of the software development process allows applying novel analysis techniques on the data and using the results to guide the process's optimization. Some researchers described various data sources and discuss the principles and techniques of data mining as applied on software engineering data. Data that can be mined is generated by most parts of the development process: requirements elicitation, development analysis, testing, debugging, and maintenance. Based on this classification they survey the mining approaches that have been used and categorize them according to the corresponding parts of the development process and the task they assist. Thus the survey provides researchers with a concise overview of data mining techniques applied to software engineering data, and aids practitioners on the selection of appropriate data mining techniques for their work.

A typical software development process has several stages; each with its own significance and dependency on the other. Each stage is often complex and generates a wide variety of data. Using data mining techniques, it can uncover hidden patterns from this data, measure the impact of each stage on the other and gather useful information to improve the software development process. The insights gained from the extracted knowledge patterns can help software engineers to predict, plan and comprehend the various intricacies of the project, allowing them to optimize future software development activities. As every stage in the development process entails a certain outcome or goal, it becomes crucial to select the best data mining techniques to achieve these goals efficiently. Some surveyed the available data mining techniques and proposed the most appropriate techniques for each stage of the development process. They also discuss how data mining improves the software development process in terms of time, cost, resources, reliability and maintainability.

In a Case-Based Reasoning Data mining can be used as efficient methods for effort estimation and automated testing has been investigated respectively If our software has many outstanding features but does not work properly due to lack of testing, your software is subjected to fail, so in order to test them properly, the test results could help the developer to classify them in different categories such as different process models and different types of errors in each developing life cycle phase, then by having these classified results and using data mining methods and Case-Based Reasoning, it would be easy to have the new software's properties and estimate the future test cases in order to reduce the cost of testing phase and eventually the developing cost in similar upcoming projects. To make it much more efficient, a case with different types of attributes is designed for each software which shows the behavior of it, then they evaluate any upcoming software by fining the most similar case for it from the stored cases and do the performed test cases for it. By estimating the proper set of domains for each attributes, they could increase the efficiency.

At present, with the scale expansion of computer software, only rely on manual for software development, maintenance and other work is more difficult. Data mining technology can accelerate the speed of software development, and can in many databases find valuable data.[17] Makes in-depth studies on software engineering data mining technology, and introduces the influence of data mining technology. Software engineering data mining technology is to use existing technology or new data mining algorithm in massive databases, and is the process of collecting valuable information for software developers through a series of steps, such as selection, analysis, formulation. It is a process of clear grasp and management of software development. Software developers must collect the required data, which is the practice of software development industry. To complete the work, they extracted the required data information from large amounts of data, and the process of collecting and selecting information is the process of data mining. At present, data mining technology has been widely used in software testing. [17] Introduces the relevant knowledge of data mining technique and its application in software testing.

III. IMPROVING TESTABILITY: AN ANALYSIS

Testing is a critical stage of the software development lifecycle. The aim is to release bug-free, software that won't cost you a fortune in backend running costs. Clearly, making this process more efficient and effective will save you time and effort, and in the long run, will improve your profitability. This is one of the

main drivers behind the switch to test automation. However, one important factor is often overlooked – software testability. In this blog, we will look at what software testability is and offer some tips and advice on how to improve the testability of your software.

Automated testing involves getting a computer to interact with your UI and replicate the actions of a real user. Things like selecting items on the screen, clicking on buttons, entering data in form fields, etc. The majority of test automation tools use some form of scripting to achieve this. These scripts first select an element in your UI, then perform some action on that element. (NB, in well-designed tests, this action may simply be verifying the correct element is in the correct place on screen).

Most test automation systems are based on a scripting language, such as JavaScript. JavaScript can select elements on the page in several ways. These include (in order of complexity) CSS selectors (e.g. Tag, ID, Class, attribute), DOM-specific (e.g. getElementById, getElementByName), and XPath. The problem is, with the possible exception of XPath, all these selectors can be ambiguous. This directly leads to the biggest bane of every test automation engineer's life: test maintenance. Each time you make a change to your UI, it risks changing the selectors. Even simple CSS changes can have an effect. As a result, every change will break some or all your tests, requiring your test scripts to be rewritten.

A related issue is the order in which selectors appear on the page. Scripting languages are relatively dumb. So, the first element that matches the selector will be the element it chooses. This can cause problems when your dev team decides to clean up their codebase. And again, this triggers the need for additional test maintenance and reduces software testability.

Challenges for manual testing

Manual testers have one key advantage over test automation engineers – they are human and therefore intelligent. This means that things, like restyling your site, moving elements on the page and even changing button names, shouldn't worry them. However, they still face some real issues. For a start, they will generally be performing tests in a static location. Many sites and applications rely on geolocation information, which is hard to test. Another key problem is application state. A real-life user quickly builds up a complex application state. Being able to replicate this with manual tests can be time-consuming and hard. Repeating it test-after-test is even harder.

The test data problem

One problem is common to both manual and automatic testing – test data. If you are going to test your system properly, you need suitable test data. You might just use a copy of your real customer data. However, this has problems. If your system handles sensitive data (e.g. **HIPAA** or banking data), you can't just allow anyone to have access to this data. Equally, if you have a new system you may not have any test data yet. In both these cases, you will end up having to create fake test data. That might sound easy enough, but it comes with a number of problems which we will explore later.

Improving testability

Below the system level in the testing hierarchy, improving software testability is largely about improving your code. This will involve things like adding explicit unit tests, utilizing tools that measure test coverage, code reviews, and the use of consistent code style. At the integration test stage, it involves understanding how each subsystem should function and may involve creating code to test for this. Where things get interesting is at the system testing stage.

Making your UI more testable

So, let's look at what can you do to make your UI more testable. The following list is by no means exhaustive but shows you some of the ways you can improve matters.

Better and consistent element naming

Your developers can improve software testability if they simply make sure every element in the UI is correctly, predictably and uniquely named. This is a challenge in large projects where you may have big teams of frontend engineers. It is also particularly challenging when developing UIs for different platforms.

Adding tools for testers

Manual testing will be much simpler if you build in tools specifically for this. For instance, you might make it simple for the application to adjust its apparent location. You might also create tools that make it easy to place the application into a known state.

Apart from that all the studies highlighted the importance of the test cases and how to increase more test case using data mining techniques combined with software testing. [2] can be made . This study also explored how data mining techniques are used to produce defect free software product. [4] It vows for the

importance of data mining and emphasizes on the automated modeling of testing systems [5] . Various discussion has been made on some data mining techniques that can be used to explore software engineering data to achieve high software reliability. We also find the goal of is to propose enhanced string distance and prioritization algorithm is to reorder the test cases to improve the efficiency.

The study also proposed state of the art data mining algorithm called Info-Fuzzy Network to execute complex data. Here they introduced an emerging methodology for automated regression testing of data driven software systems.

IV. CASE STUDY IN OPEN SOURCE SOFTWARE

The major goal of software development is to deliver high-quality software efficiently. There has been a huge growth in the demand for software quality during recent ages. In recent decades the production of large software projects are very large and is costly and time consuming. As a consequence, issues are related to testing, becoming increasingly critical. The ability to measure software defect can be extremely important for minimizing cost and improving the overall effectiveness of the testing process. The possibility of early estimating the probable faultiness of software could help on planning, controlling and executing software development activities. A low cost method for defect analysis is learning from past mistakes to prevent future ones. Today, there exist several data sets that could be mined in order to discover useful knowledge regarding defects.

Open source/free software (OSS/FS) is the software where users have the freedom to run, copy, distribute, study, modify and improve the software, and the source code is freely available. Briefly, OSS/FS systems are programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program. The only thing that has to be taken care is that there should be no degradation of software in terms of productivity and quality. Open Source Software (OSS) project managers often need to observe project key indicators, e.g., how much efforts are needed to finish certain tasks, to assess and improve project and product quality, e.g., by analyzing defect data from OSS project developer activities and with the expansion of software size and complexity, how to detect defects becomes a challenging problem.

Data mining techniques are applied in building software fault prediction models for improving the software quality. Early identification of high-risk modules can assist in quality enhancement efforts to modules that are likely to have a high number of faults. Classification tree models are simple and effective as software quality prediction models, and timely predictions of defects from such models can be used to achieve high software reliability.

V. CONCLUSION

Nowadays data mining techniques are extensively used in software testing, and applying data mining in software testing can significantly increase the efficiency of software system to a certain extent. Data mining techniques can be used in software testing to produce high quality software. In this paper it is proved that one can improve software quality through software defect prediction using a variety of data mining techniques.

REFERENCES

[1]. Emad Alsukhni, Ahmad A. Saifan et al, "A New Data Mining-Based Framework to Test Case Prioritization Using Software Defect Prediction", International Journal of Open Source Software and Processes 8(1):21-41 · January 2017.

[2]. Yanguang Shen, Jie Liu et al., "Research on the Application of Data Mining in Software Testing and Defects Analysis", 2009 Second International Conference on Intelligent Computation Technology and Automation, DOI: 10.1109/ICICTA.2009.384, Print ISBN: 978-0-7695-3804-4, IEEE.

[3]. A.Kanagaraj, Dr Antony Selvadoss Thanamani, "A Study on Technologies Used in Ubiquitous and Pervasive Computing", International Journal of Advanced Research in Computer Science, Volume 3, No.3, May-June 2012.

[4]. Last M., "Data Mining for Software Testing", In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA, 2005.

[5]. Nadhem Sultan Ali, Dr. V.P. Pawar, "The use of data Mining Techniques for Improving Software Reliability", International Journal of Advanced Research in Computer Science, Volume 4, No. 4, March-April 2013.

[6]. Halim. S., Jawawi, D. N., & Sahak. M., "Similarity distance measure and prioritization algorithm for test case prioritization in software product line testing", Journal of Information and Communication Technology, 18(1), 57-75, 2018.

[7]. Mark Last, "Using Data Mining For Automated Design of Software Tests", Department of Information Systems Engineering, Ben-Gurion University of the Negev.

[8]. M. Last, M. Friedman et al., "Using Data Mining For Automated Software Testing", International Journal of Software Engineering and Knowledge Engineering, Vol. 14, No. 4 (2004) 369-393, World Scientific Publishing Company.

[9]. Dr A.Kanagaraj, S.Sharmila, "Pervasive Computing Based Intelligent Energy Conservation System", Int. J. Advanced Networking and Applications, Volume: 07 Issue: 03 Pages: 2736-2740 (2015) ISSN: 0975-0290, IJANA.

[10]. Amit Kumar Jakhar and Kumar Rajnish, "Software Fault Prediction with Data Mining Techniques by Using Feature Selection Based Models", International Journal on Electrical Engineering and Informatics - Volume 10, Number 3, September 2018.

[11]. Tao Xie, Jian Pei et al., "Mining Software Engineering Data", 29th International Conference on Software Engineering (ICSE'07 Companion), IEEE, Print ISBN: 0- 7695-2892-9, 20-26 May 2007, Minneapolis, MN, USA.

[12]. Mark Last, Menahem Friedman et al., "The data mining approach to automated software testing", SIGKDD '03, August 24-27, 2003, Washington, DC, USA, 2003 ACM 1-58113-737-0/03/0008.
[13]. Mariam Bibi, Rubab Mehboob et al., "Analytical Study of Data Mining Techniques for Software Quality Assurance", International Journal of Computer and Communication System Engineering (IJCCSE), Vol. 2 (3), 2015, 377-386, ISSN: 2312-7694.
[14]. M. Halkidi, D. Spinellis et al., "Data mining in software engineering", Intelligent Data Analysis 15 (2011) 413–441, DOI 10.3233/IDA20100475, IOS Press.
[15]. Nidhin Thomas, Atharva Joshi et al., "Data Mining Techniques used in Software Engineering: A Survey", International Journal of Computer Sciences and Engineering, Volume-4, Issue-3, E-ISSN: 2347-2693, 2015, pp.28-34.
[16]. Ali Ilkhani and Golnoosh Abaee, "Extracting Test Cases by Using Data Mining; Reducing the Cost of Testing", International Journal of Computer Information