Improvement of Software Reliability using Data Mining Technique

Amitava Bondyopadhyay[1] , Dr. A.C Mandal[2]

[1]Department of Computer Science, Mankar College
[2]Department of Computer Science, The University of Burdwan

DOI: 10.29322/IJSRP.12.06.2022.p126XX
http://dx.doi.org/10.29322/IJSRP.12.06.2022.p126XX

Paper Received Date: 20th May 2022
Paper Acceptance Date: 5th June 2022
Paper Publication Date: 14th June 2022

Abstract- There are some possible steps that can be used to improve Software reliability and robustness through Data Mining. Improving Software Reliability and Robustness would ultimately improve software quality. Defect prediction for a software system is a technique which predicts defects from historical database. This technique helps to enhance the software reliability.

Index Terms- Cost, Classification, Data mining, Database, Normalization, Dataset, Testing

I. Introduction

1. Introduction to Software Quality [2]

By software quality we mean, some software which are practically error free and that are made in time and follows a particular financial plan, implements the requirements and are maintainable. While talking about software engineering background, it refers to both functional plus structural quality. Software Functional Quality tells whether it meets a given design according to requirements specifications. Software Structural Quality fulfills non-functional requirements.

1.1 Challenges of the Quality of a Software Product

A software is never be of free of defects for the following causes.

1.1.1 Complexity of Software Product

Generally software have thousands of operational potential. So making of all these operational options correctly becomes the most significant challenge a software industry.

1.1.2 Visibility of the Product

As the software is visible to the programmer, so major portion of the errors can be traced during the development phase.

1.2 Software Development Process

While building software, defects can be detected during phases given below.

- **While developing the Software** – In the Software industry the designers, developers verifies the product prototype to detect defects, if any.
- **While making the planning of Software product** – Since in this phase, the software's are designed, so one can easily examine the product to trace the defects which are ignored during the development phase.
- **During the manufacturing process of Software** –While implementing the quality assurance, defects in the product, detected earlier can usually be corrected by chaining design of the software product in a way that removes such defects manufactured in future.

Since defects in the software that can be found is in the product development phase, so product manufacture planning and manufacturing phases are not needed because production of software modules and writing of software manuals are made automatically.

1.3 Factors that have an effect on Software Quality

There are several factors that have a direct effect on software. Basically they are of two types. First factors are those which can be calculated directly like the no. of logical error. The 2nd factor can not be measured directly. It may be the factor of quality control. There are several models of quality factors suggested over the years. From those models McCall's model provides realistic and up to date techniques for classifying requirements of software.

1.3.1 Defining McCall's Factor Model [5]

This model categories software requirement into eleven separate quality factors. They are grouped in the following way.

Correctness: It tells us whether the software is error free or not.
Reliability: It is the probability of how much time software remains failure free.
Efficiency: It is the amount of software developed divided by number of resources.
Integrity: It refers to the quality of the software code.
Usability: It refers to the degree to which a software can be used by the customer and get satisfaction
Maintainability: It refers to the degree at which a software application is repaired or enhanced.

Flexibility: It means how easily software can adapt to a newer requirement.
Testability: It refers to how easily a fault can be traced.
Portability: It means how easily software can adapt to a newer platform.
Reusability: Whether software can be reused for some specific purpose.
Interoperability: It indicates the ability of different software to communicate with each other

1.4 Introduction to Data mining

Data mining[9], is a method of detecting patterns and important information from big data sets. Since the growth of big data analysis, evolution of data warehousing technology acceptance of data mining methods has quickly accelerated over the last ten years. Although the technology endlessly evolves to process data at large-scale, researchers faces challenges with its automation. Data mining improves decision making ability. It can target dataset, predicts outcome by using machine learning techniques., Those methods can filter data, can detect fraud or security breaches.

The extraction of information from large amounts of data - can be viewed as a natural result of evolution of information technology. The early decades witnessed a lot of focus on data collection followed by an upsurge in the interest on data management. After this, the focus has shifted to advanced data analysis tasks. According to Han and Kamber (2006)[10] , the profusion of data coupled with the requirement for advanced analysis tools has created a "data rich, information poor" situation.

Software engineering is primarily concerned with the development of quality software on time and within the cost estimates. The broad applicability of data mining for a variety of disciplines has raised questions about its applicability to the domain of software engineering. The discipline of Software Engineering is fraught with multiple challenges and there have been attempts in this regard. The present study is aimed at presenting a brief overview of some of these works that attempt to address the challenge of improving software quality.

Hayes et al.,(2005)[11] state that there are two ways by which data mining can be applied to software engineering. The first is in exploratory studies of existing artifacts like source code, test case documentation and the like to find out novel patterns. The second is for the improvement and automation of SDLC processes. This can pick up the speed of performance of many activities but this is not error-tolerant. So in this case, the results are presented to the analyst who assesses the correctness and gives the final results.
Wahidah Husain et al (2011) categorize[22] software engineering data into:

- Sequences like execution traces collected dynamically. For example, the method call data.
- Graphs like dynamic call graphs coming out from the source code.
- Texts like code comments and documentation.

Mining of sequential data can be helpful in flaw or bug detection. We have Frequent item-set mining and frequent sequence mining for this purpose. Most software engineering data can be conveniently expressed as a graph, and for this reason graph data mining is an lively area of research in SE data mining. Mining of software behavior graphs collected from program execution can be used to disclose traces of bugs. According to Gegick et al., (2010) 80% of information in computers is stored as text. In the context of software engineering, useful text data include software requirements specification, bug reports and the like.

1.5 Mining software repositories[20]

Hassan (2008) states that software repositories have massive information and classifies repositories into following:

- Based on previous data, the historical repositories like source code, bug reports, and communications pertaining to a project.
- Dynamic repositories such as deployment logs containing details about the execution of a particular application at a particular operational site.
- Code repositories like google code which contain source code developed by various developers.

According to Hassan (2008) notwithstanding the availability of repositories, the data available in these repositories is not in the plan of software engineering researchers due to the following reasons:

- Access Limitations – The reluctance of software organizations to grant access to repositories pertaining to their software to researchers is the main road block. Available data pertain to academic projects which are small and uninteresting at times.
- Data Extraction Difficulties – As the repositories are not built keeping in mind the requirement of automated extraction, they tend become difficult to mine.

1.5.1 CHALLENGES IN MINING SOFTWARE ENGINEERING DATA

Xie et al., (2012) state the following as challenges in mining SE data.

- Requirements Unique to SE – Most available data mining tools are general purpose, and applying these tools to mine software engineering data undermines the requirements unique to software engineering., On the one hand, software practitioners lack the ability to modify data mining algorithms to tune them for software engineering, and on the other hand, data mining researchers lack the knowledge about software engineering preventing them from developing tools fit for SE data.
- Complex Data and Patterns – Existing data mining algorithms may not be able to produce desired pattern representations in the context of SE. In many cases, there is a need to mine multiple kinds of data together.
- Large Scale Data – Dynamic traces produced from a simple sized program can be huge, thereby challenging the abilities of data mining techniques.
- Defining Just-In-Time Mining – According to Xie et al., (2012) provision of rapid feedback is the

necessity of the day and for this software, developers must be capable to mine SE data on the fly.

1.5.2 USES OF DATA MINING IN SOFTWARE ENGINEERING[1]

Xie et al., (2009) presented a comprehensive overview of applying data mining to software engineering. They state that SE data concern 3P's – People, Processes and Products. They categorize SE data into:

- Sequences – such as dynamic traces
- Graphs – like the dynamic call graphs generated by execution
- Texts– such as bug reports and email documents.

According to Xie et al., (2009) the main steps in mining SE data are:

Collect/Investigate SE data – Here software developers can adopt a problem-driven approach (like selecting what SE tasks to assist) or data-driven approach (knowing what SE data to mine).
Determine SE task – Here the software engineers find out which particular SE task can be benefited from mining.

- Preprocessing – This entails extracting the relevant information from raw SE data. This can be static method call sequences obtained from the source code or execution traces produced dynamically. The data can be further cleaned to make it suitable for a particular data mining algorithm.
- Adapt/Adopt/Develop Mining Algorithm – Mining algorithms can be classified into: Clustering and Classification, Frequent Pattern Mining, and Pattern Matching.
- Post processing – The results of the mining algorithm are converted into a format required for the particular SE task of non-availability of real databases with information on parameters conditioning the development of many software projects, in the present scenario the problem is largely addressed by the existence of powerful simulation systems.
  - **Causes of software failure**

By a software system we mean a product that helps business domain. But in today's world, in spite of having a lot of experiences, software failure has become a regular feature. Those failure occur due to fault happened in SDLC or due to lack of requirement specification by the customer.

1.6 Application of Machine Learning in Software Defect Prediction

Machine learning is one of the fastest growing fields nowadays. It produces various learning algorithms for different application. Among those some of the learning algorithms are beneficial to software engineering. More ever, different machine learning researchers are proposing different algorithm for software fault prediction . From that, we can classify high quality defect models into those that are based on classification, clustering and ensemble methods.

1.7 Machine Learning in Software Defect Prediction [19]

One of the good research fields in today's world is Machine learning. Its purpose is to produce different learning methods for different application. These applications are very much helpful in software engineering. Researchers are proposing different methods of learning very now and then. With the help of those algorithms we can easily classify our problems as per needs.

II. Introduction to Data mining

Data mining[59], is a method of detecting patterns and important information from big data sets. Since the growth of big data analysis, evolution of data warehousing technology acceptance of data mining methods has quickly accelerated over the last ten years. Although the technology endlessly evolves to process data at large-scale, researchers faces challenges with its automation. Data mining improves decision making ability. It can target dataset, predicts outcome by using machine learning techniques., Those methods can filter data, can detect fraud or security breaches.

The extraction of information from large amounts of data - can be viewed as a natural result of evolution of information technology. The early decades witnessed a lot of focus on data collection followed by an upsurge in the interest on data management. After this, the focus has shifted to advanced data analysis tasks. According to Han and Kamber (2006)[9] , the profusion of data coupled with the requirement for advanced analysis tools has created a "data rich, information poor" situation.

Software engineering is primarily concerned with the development of quality software on time and within the cost estimates. The broad applicability of data mining for a variety of disciplines has raised questions about its applicability to the domain of software engineering. The discipline of Software Engineering is fraught with multiple challenges and there have been attempts in this regard. The present study is aimed at presenting a brief overview of some of these works that attempt to address the challenge of improving software quality.

III. Study of improving Software Reliability and Robustness through Data Mining

Historical data has some hidden parts that are very helpful and well-informed and can be used to predict, plan and recognize various aspects of a software project. We can get meaningful information from those data using data mining techniques. With those techniques one can mine common patterns and identify breach of patterns. Hence by this way data can be converted into knowledge and we can be capable to get defect detection, testing, debugging and maintenance which will make highly productive and reliable software.

We always want to make good quality software efficiently. But as the day passes making of quality of software become difficult. Especially if the project is a bigger one. If we want to increase the quality of our software system we have to increase the efficiency of software testing .So we need to find defects from the software modules at the beginning. An early estimation of fault will give much help on software building process. Always we should want a low cost mechanism that learns from previous error to prevent future one. In recent research, we would be able to find a few data set that can be mined to find helpfu facts about the defects. Hence the aim should be to identify high risk modules.

3.1 Review of Literature

By using different types of data mining techniques like classification, association and clustering, we can be able to mine different software engineering data needed for research purpose Those data can be used in defect detection, testing, debugging and maintenance. Rules and patterns are usually mined from source code. There are several technique for that purpose. Some of them are Statistical technique, PR-Miner technique, Frequent Item set Mining. Following flowchart gives details of PR-Miner technique:-

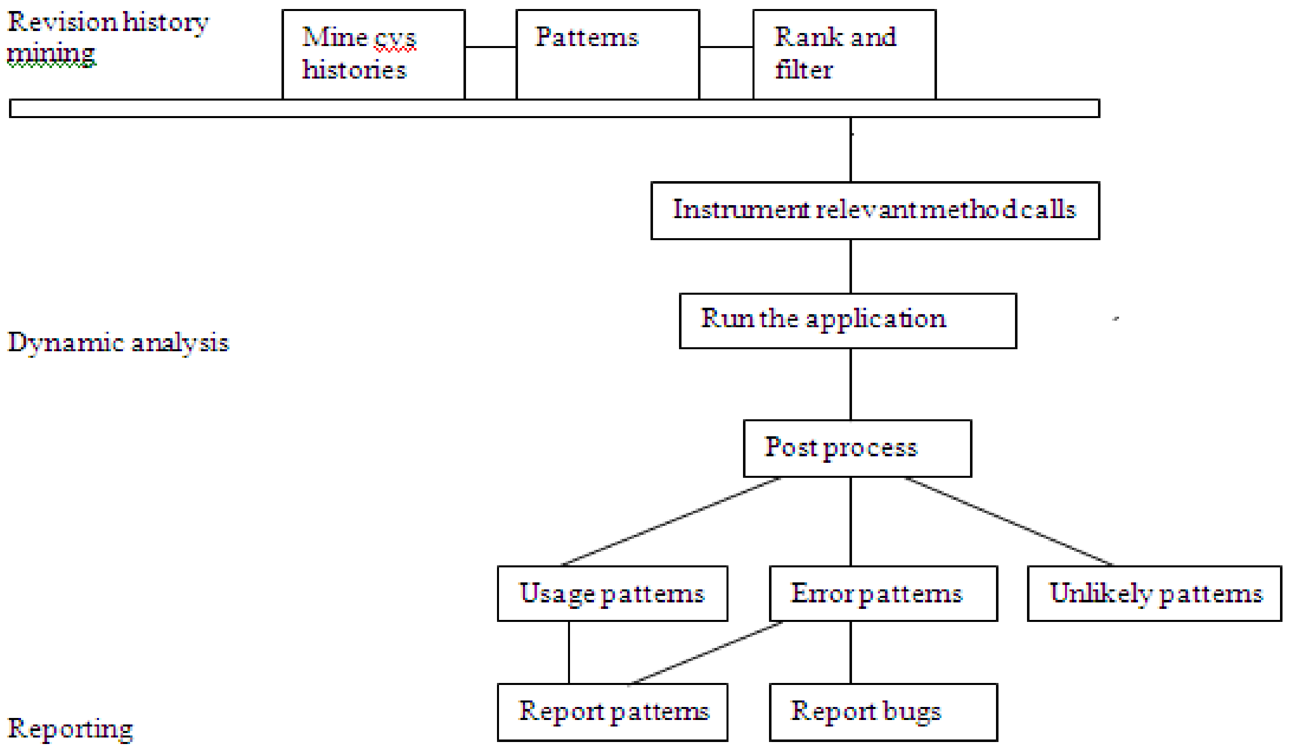

Figure 3.1: Data-Mining Strategy

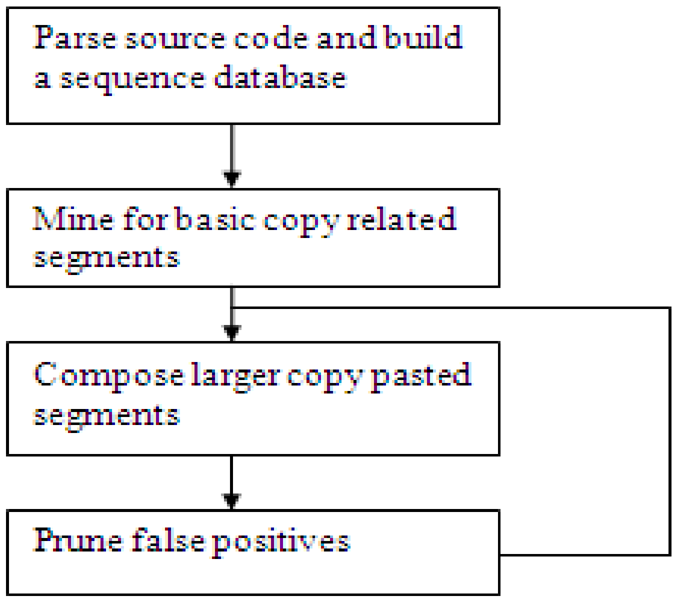
Parse source code and build a sequence database
Mine for basic copy related segments
Compose larger copy pasted segments
Prune false positives

Fig3 CP Mining

3.2 Study of Improving Software Robustness [4]

Generally a program falters mainly for in two way: logic errors in the code, and exception failures. Two-third of system crashes [6] are for exception failures. In order to decrease exception failures. code reviews, walkthroughs and formal testing are made. But the core problem remains the same- programmer’s inadequate coverage of exception conditions that ultimately reduces the robustness of a software. Exceptions are runtime error. It occurs when a program is barred by existing circumstances from providing its particular service. It could be an arithmetic exception(division by zero). Exception can be caused by many different conditions. Some of them are empty data file, insufficient memory, type mismatch, wrong command- line argument, protection violation and bad data returned from another program. Hence decreasing the occurrence of failure can be beneficial. It would ultimately increases the robustness of a software. Data mining techniques can be applied to increases the robustness. Dependability cases can be beneficial for this purpose.

IV. Conclusion

This paper discusses the possible steps that can be used to improve Software Reliability and Robustness through Data Mining. Improving Software Reliability and Robustness would ultimately improve software quality.

References

[1] K. O. Elish and M. O. Elish, ―Predicting defect-prone software modules using support vector machines,‖ J. Syst. Softw., vol. 81, no. 5, pp. 649– 660, 2008.

[2] I. Gondra, Applying machine learning to software fault-proneness prediction,‖ J. Syst. Softw., vol. 81, no. 2, pp. 186–195, 2008.

[3] J. Zheng, Cost-sensitive boosting neural networks for software defect prediction,‖ Expert Syst. Appl., vol. 37, no. 6, pp. 4537–4543, 201.

[4] NASA Software Defect Datasets [Online]. Available: https://nasa softwaredefectdatasets.wikispaces.com. [Accessed: 01-April-2019].

[5] NASA Defect Dataset.‖ [Online]. Available: https://github.com/klainfo/NASADefectDataset. [Accessed: 01-April-2019].

[6] Mudhu Sudhan Chakraborty, Amitava Bondyopadhyay and Tapas Kumar Ghosh,Towards more efficient 3NF determination using reduced functional dependency sets, Advances and Applications in Mathematical Sciences,Mili Publication,2021

[7] Ms. Puneet Jai Kaur, Ms. Pallavi, Data Mining Techniques for Software Defect Prediction, International Journal of Software and Web Sciences (IJSWS)

[8] Wahidah Husain, Pey Ven Low, Lee Koon Ng, Zhen Li Ong, Application of Data Mining Techniques for Improving Software Engineering, ICIT 2011 The 5th International Conference on Information Technology

[9] K.B.S Sastry, Dr.B.V.Subba Rao, Dr K.V.Sambasiva Rao, Software Defect Prediction from Historical Data, International Journal of Advanced Research in Computer Science and Software Engineering.

[10] Tao Xie, Suresh Thummalapenta, David Lo, and Chao Liu. Data mining for software engineering. *Computer*, 42(8):55–62, 2009.

[11] Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu. A general software defect-proneness prediction framework. *Software Engineering, IEEE Transactions on*, 37(3):356–370, 2011.

[12] Ma Baojun, Karel Dejaeger, Jan Vanthienen, and Bart Baesens. Software defect prediction based on association rule classification. *Available at SSRN 1785381*, 2011.

[13] S Bibi, G Tsoumakas, I Stamelos, and I Vlahavas. Software defect prediction using regression via classification. In *IEEE International Conference on*, pages 330–336, 2006.

[14] Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. *Software Engineering, IEEE Transactions on*, 33(1):2–13, 2007.

[15] Iker Gondra. Applying machine learning to software fault-proneness prediction. *Journal of Systems and Software*, 81(2):186–195, 2008.

[16] Ata¸c Deniz Oral and Ay¸se Ba¸sar Bener. Defect prediction for embedded software. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1–6. IEEE, 2007.

Authors

First Author – Amitava Bondyopadhyay, Department of Computer Science, Mankar College
Second Author – Dr. A.C Mandal, Department of Computer Science, The University of Burdwan