Role of Data Mining in Improving Software Development Process

S.Jeya Priya[1], C.Rekha[2]

[1]*Research Scholar, Madurai Kamaraj University, India*
[2]*Department of Computer Science, Government Arts College, Melur, India*

Abstract

Software engineering data (such as code bases, execution traces, historical code changes, mailing lists, and bug databases) 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. This paper explores how various data mining techniques can be applied for software development and its outcomes. Every step in software engineering techniques needs different mining techniques and strategies. The author highlights the core of the relationship between the software engineering and data mining in this paper.

Keywords: Classification, Clustering, Mining Techniques, Software Reusability, Data Mining, Software Development.

I. INTRODUCTION

In Computer Science, software engineering deals with the design and creation of programs for computers and other electronic devices. [1] In a typical software development methodology, the work is split into distinct stages with specific activities in each, with the intention of improvingplanning and management [2]. Another side Data mining is defined as the process of discovering previously unknown and potentially useful information from data collections. Thus utilizing data mining in software engineering with the aim of software improvement has piqued the interest of researchers worldwide. There are several challenges that emerge in mining software repositories [3]. The major ones being, dealing with the inherent complexity and sheer volume of the software engineering data.

II. STATE-OF-THE-ART

Perhaps the earliest survey of the use of data mining in software engineering is the1999 Data and Analysis Center for Software (DACS) state-of-the-art report [4]. It consists of a thorough survey of data mining techniques, withemphasis on applications to software engineering, including a list of 55 data miningproducts with detailed descriptions of each product and summary information along anumber of technical as well as process-dependent features.Since then, and over the years, Xie[5] has been compiling and maintainingan

(almost exhaustive) online bibliography on mining software engineering data. Healso presented tutorials on that subject at the International onference on KnowledgeDiscovery in Databases in 2006 and at the International Conference on SoftwareEngineering in 2007, 2008 and 2009.

Clarke [6] stated that the maintainability of software becomes difficult with the increase in its complexity. This might eventually result in problems regarding software integrity and bug detection. A bug is a flaw in a computer program that can ultimately cause glitches, program failure or software destruction.Kagdi et al. [7] have recently published a detailed taxonomy of software evolution data mining methodologies and identifies a number of related research issues that require further investigation.

Taylor et al. [8] have produced a comprehensive survey covering the most recent applications of data mining to software engineering. They also discuss the issues one might encounter in mining software data and the necessary conditions for success. Halkidi et al. [9] have written one of the most thorough papers on the subject. Their work features in depths look at the data mining techniques and how they can be effectively applied in software engineering.

Aouf et al. [10] described how clustering techniques could be used to discover hidden patterns in data to gather valuable information. Chang and Chu [11] showed how Association rule mining could be used to detect software defects. Gegick et al. [12] demonstrated the importance and usage of text mining in bug identification whereas Runeson et al. [13] showcased the capabilities of NLP in tackling duplicate defect reports. Islam and Brankovic [14] proposed techniques to ensure privacy in data mining. This was mainly done by filling parts of the dataset with noisy data. On the other hand, Ma and Chan [15] in their work suggested iterative mining for mining overlapping patterns in noisy data. While, Islam and Brankovic [14] were concerned with preserving privacy with the help of noisy data, Ma and Chan [15] dealt with the elimination of noisy data to achieve the objective of extracting valuable information.

Taghi.M et al [16] presents a study on the use of CART (a classification tree algorithm) to identify fault prone software modules based on product and process metrics. The data is drawn from large telecommunication software systems at Nortel.

Andy Podgurski et al [17] present a methodology to estimate operational software reliability by stratified sample of beta testers' code execution profiles. Cluster analysis is used to group

code executions into dissimilar profiles. The authors show that more accurate estimates of failure frequencies can be drawn by stratified samples of those clustered execution profiles.

Norman Fenton and Martin Neil [18] explore the use of Bayesian Belief Networks (BBN) to build defect prediction models. These are the preliminary results of an interesting work. The paper has a wonderful discussion on the limitation of traditional defect prediction models. The authors argue that BBN models are interpretable and can include contextual software process information in them. This allows domain experts to analyze how defect introduction and detection variables affect the defect density counts in the model.

The papers [19-21] proposed a model that has the same procedure in reducing the test cases using k-means algorithm based on path coverage. In [22], they used the branch coverage. The model starts by identifying a centroid for each cluster. Then, they take each test case and associate it to the nearest centroid. After that, they recalculate the new centroid for each cluster. However, they verified their approach using a simple system that contains two variables. It means it will be so hard to apply this technique with more complex systems. Moreover, in their approach they identify the number of clusters in advance (i.e. before the reduction process started) based on the prime paths generated from the code.

Other researchers have worked in the test case reduction field to improve systems testing processes, such as regression testing [23]. In order to find a more efficient algorithm for reducing test cases, they have presented an approach that assigns priority for each test case. Priority is given depending upon the code coverage, and higher priority test case value is selected for the reduced test suites. To demonstrate the effectiveness of their algorithm, the approach is applied to two applications.

III. Mining Techniques for Development Phases

In the Requirements Elicitation phase, the requirement document provides a full description of all the software and hardware requirements for the projects. Since this document is highly detailed and descriptive in nature, it increases the time required to summarize the requirements in such a way that they are available at the appropriate time. Time management in resource availability is critical for the functioning of all the subsequent phases. Data mining techniques such as classification could be used on such data, which will classify and prioritize the requirements in such a way that all the resources which are required at each stage shall be present on time. Text mining can be used to summarize the huge amount of given data. This will decrease the amount

man hours put into summarizing and prioritizing the requirements, thereby saving time, cost and human resources.

In the design phase, while designing the layout of the architecture and planning out the database structure it becomes critical to know which data would be required where and when. Data mining techniques such as Clustering can gather similar data from time to time so that extraction of data will become easier. Data gathering becomes a tedious job especially when it has to be pre-processed over and over again. By using Clustering on data elements, the data can be differentiated based on its similarity or dissimilarity. Labeling data from any incoming site would also be much easier using clustering.

During implementation, independent parts of codes or modules are implemented first, after which they are integrated with each other. This integration phase can prove to be more challenging than actually coding these modules. The functionalities of each module has to be understood so that they can be integrated efficiently. Data mining techniques such as classification and text mining will allow the developer to understand the possible bugs that might occur during integration. Here the input would be the source code of these independent modules and the output would be whether or not there would be bugs after integration.

Frequent pattern mining will also help in correcting those defects that are discovered while performing classification. Clustering can help group together the software processes that are similar. The reliability of a software system is inversely proportional to the number of failures and bugs encountered in the software. Using the data mining techniques mentioned above, these bugs and failures can be detected easily and rectified. This saves valuable time, money and the additional resources that might have been required for their detection and resolution along with increasing the reliability and maintainability of the software.

While testing the software, unit testing will usually be performed at the implementation level. However, theother testing techniques will most likely be performed bya tester who isn't from the development team. The job of finding bugs in a code is time consuming and the possible test cases are infinite. Classification techniques can be usedfor I/O variables of the system which will produce a network with sets for functional testing. This reduces the time for testing phase and the product can be released as early as possible.

As a result, it also prevents the development time from extending which in turns saves cost and resources. The tester has to also look out for behavioral/state changes in the execution of the program. Clustering and classification techniques can be used to look out for such changesin the system behavior for a particular set of data of testing.

Table 1. Development stages with mining techniques

Software Development Stage	Data Mining Techniques	Input Data	Data Analysis Result
Requirement Elicitation	Classification	Documentation	Classification of requirements
	Text mining	Mailing lists	Data Summarization
Design	Clustering	Design document	Data gathering, labeling
Implementation	Clustering	Source code	Software processes
	Classification	SCM	Bug tracking
	Text	Mining Source code	Bug tracking
	Frequent Pattern Mining & Association rules	Defect	Defect Correction
		Program dependence graph	Neglected Conditions
Testing	Classification	I/O variables of software system	A network producing sets for function testing
		Program executions	Software behavior classifiers
	Clustering	Execution Profiles	Clusters of execution profiles

IV. CONCLUSION

Software engineering handling huge amount of structured and un-structures data for every development phase. Established the need and importance of using data mining techniques to aid software engineering, especially to tackle problems such as the occurrence of bugs, rise in the cost of software maintenance; unclear requirements, etc. that can affect software productivity and quality. This paper highlights the usage of mining algorithms in the development phase and discussed the details analysis of each development phase and its outcomes.

REFERENCE

[1] Laplante, Phillip (2007). "What Every Engineer Should Know about Software Engineering", Boca Raton: CRC.

[2] "Selecting a development approach", Centers for Medicare & Medicaid Services (CMS) Office of Information Service (2008). Re-validated: March 27, 2008.Retrieved 27 Oct 2015.

[3] T. Xie, S. Thummalapenta, D. Lo and C. Liu, "Data mining for software engineering", IEEE Computer Society, Volume 42, Issue 08, Page No (55-62), August 2009.

[4] Sunderhaft, N.L. (1999). Mining Software Engineering Data : A Survey A DACS State-ofthe-Art Report.

[5] Xie, T. (2010) 'Bibliography on mining software engineering data', available athttp://ase.csc.ncsu.edu/dmse.

[6] J. Clarke et al., "Refomulating software engineer as a search problem," IEEE Proceeding Software., Volume 150, Issue 03, Page No (161-175), June 2003.

[7] Kagdi, H., Collard, M.L. and Maletic, J.I. "A survey and taxonomy of approaches for mining software repositories in the context of software evolution", Journal of Software Maintenance and Evolution: Research and Practice, Volume 19, Issue 02, Page No (77–131).

[8] Taylor, Q.and Giraud-Carrier, C. "Applications of data mining in software engineering", International Journal of Data Analysis Techniques and Strategies, Volume 02, Issue 03, Page No (243-257), July 2010.

[9] M. Halkidia, D. Spinellisb, G. Tsatsaronisc and M.Vazirgiannis, "Data mining in software engineering", Intelligent Data Analysis 15, Page No (413–441), 2011.

[10] M. Aouf, L. Lyanage, and S. Hansen, "Critical review of data mining techniques for gene expression analysis," International Conference on Information and Automation for Sustainability (ICIAFS) 2008, Page No (367-371), 2008.

[11] C. CHANG and C. CHU, "Software Defect Prediction Using Inter transaction Association Rule Mining", International Journal of Software Engineering and Knowledge Engineering, Volume 19, Issue 06, Page No (747-764), September 2009.

[12] M. Gegick, P. Rotella and T. Xie, "Identifying security bug reports via text mining: an industrial case study", Mining Software Repositories (MSR), 7th IEEE Working Conference, Page No (11 – 20), 2010.

[13] P. Runeson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing", Software Engineering, 2007. ICSE 2007.29th International Conference, Page No (499 – 510), 2007.

[14] M. Z. Islam and L. Brankovic, "Detective: a decision tree based categorical value clustering and perturbation technique for preserving privacy in data mining," Third IEEE Conference on Industrial Informatics (INDIN), Page No (701-708), 2005.

[15] P. C. H. Ma and K. C. C. Chan, "An iterative data mining approach for mining overlapping coexpression patterns in noisy gene expression data," IEEE Trans. NanoBioscience, Volume 08, Issue 03, Page No (252-258), September 2009.

[16]Taghi M. Khoshgoftaar and Edward B. Allen. Modeling Software Quality with Classification Trees.In Recent Advances in Reliability and Quality Engineering, Hoang Pham Editor.World Scientific, Singapore, 1999.

[17] Andy Podgurski, WassimMasri, Yolanda McCleese, and Francis G. Wolff. Estimation of Software Reliability by

[18] Norman Fenton and Martin Neil.A Critique of Software Defect Prediction Models.To appear in the IEEE Trans. on Soft.Eng., 1999.Stephen G. Eick, AudrisMockus, Tood L. Graves, Alan F. Karr.A Web Laboratory for Software Data Analysis.World Wide Web, 12, pp. 55-60, 1998.

[19] Muthyala, K., & Naidu, R. (2011). A novel approach to test suite reduction using data mining. Indian Journal of Computer Science and Engineering, 2(3), 500-505.

[20] Kameswari, U. J., Saikiran, A., Reddy, K. V. K., &Varun, N. Novel Techniques For Test Suite Reduction. International Journal of Science and Advanced Technology, 1(8).

[21] Dash, R., && Dash, R. (2012). Application of K-mean Algorithm in Software maintenance. International Journal of Emerging Technology and Advanced Engineering,Vol. 2(5).

[22] Chantrapornchai, C., Kinputtan, K., &Santibowanwing, A. (2014). Test Case Reduction Case Study for White Box Testing and Black Box Testing using Data Mining. International Journal of Software Engineering and Its Applications, 8(6), 319-338.

[23] Pravin, A., &Srinivasan, D. S. (2013). An Efficient Algorithm for reducing the test cases which is used for performing regression testing. In 2nd International Conference on Computational Techniques and Artificial Intelligence, Dubai (UAE) (pp. 194-197).