ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

A Survey on the Application of Data Mining Techniques for Software Quality Enhancement

A. R. Visagan[1], Dr. M. Sumathi[2], Dr. G. Sujatha[3]
Research Scholar, Madurai Kamaraj University, Madurai, India[1]
Associate Professor, Sri Meenakshi Govt. Arts College for Women, Madurai, India[2]
Associate Professor, Sri Meenakshi Govt. Arts College for Women, Madurai, India[3]

Abstract: Software Engineering is about the development of quality software. Quality of software is a multi-faceted concept and software engineering researchers have sought to develop techniques that can enhance software quality. Data Mining has seen a huge success in various other disciplines and it is natural to contemplate its applicability to address Software Engineering challenges. Several works have been reported in the literature and these works throw light on the prospects of applying data mining techniques to address hitherto unaddressed challenges in the endeavor to develop quality software. The present paper attempts to give an insight in to some of the reported works in the literature in this regard.

Keywords: Data Mining, Software Quality, Software Engineering, Software Quality Enhancement

I. INTRODUCTION

Data mining – the extraction of knowledge 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), the abundance of data coupled with the need for advanced analysis tools has created a "data rich, information poor" situation.

Software engineering is the application of systematic, disciplined and quantifiable approach to the design, development, operation and maintenance of software. It had its origins in the late 1960's and was primarily aimed at addressing the issue then described as "Software Crisis". 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) 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 software life cycle processes. This can improve 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 software engineering data into:

- Sequences such as execution traces collected at run time. For example, the method call data.
- Graphs such as dynamic call graphs generated from the source code.
- Texts such as code comments and documentation.

Mining of sequential data can be helpful in flaw or bug detection. Frequent item-set mining and frequent sequence mining can be very useful for this purpose.

Most software engineering data can be conveniently expressed as a graph, and for this reason graph data mining is an active area of research in SE data mining. Mining of software behavior graphs

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

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. As an example of applying text mining to software engineering, Wahidah Husain et al.,(2011) provide an example of applying text mining to classify bug reports (which contain text data) into Security Bug Reports(SBRs) and Non-Security Bug Reports(NSBRs).

The rest of the paper is organized as follows: Sections 2, 3 and 4 present a broad picture of applicability of Data Mining for Software Engineering in general. Subsequent sections explore some of the reported works that specifically tackle the quality issue by application of data mining.

II. Mining Software Repositories

Hassan (2008) states that software repositories contain a wealth of information and classifies repositories into:

- Historical repositories such as source code, bug reports, and communications pertaining to a project.
- Run-time repositories such as deployment logs containing information about the execution of a particular application at a particular deployment site.
- Code repositories such as 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 have not been the focus 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.

III. 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 – Execution traces produced from an even average sized program can be huge, thereby challenging the abilities of data mining algorithms.
- Just-In-Time Mining – According to Xie et al., (2012) provision of rapid feedback is the necessity of the day and for this software, engineers must be able to mine SE data on the fly.

IV. Example Applications Of Data Mining For Software Engineering

Xie et al., (2009) presented acomprehensive 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 execution traces
- Graphs – such as 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:

1. Collect/Investigate SE data – Here software engineers can adopt a problem-driven approach (like selecting what SE tasks to

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

assist) or data-driven approach (knowing what SE data to mine).

2. 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: Frequent Pattern Mining, Pattern Matching, Clustering and Classification.
- Post processing – The results of the mining algorithm are converted into a format required for the particular SE task.

Xie et al., (2009) given the following example applications of applying data mining to software engineering.

TABLE 1
EXAMPLE APPLICATIONS OF DATA MINING FOR SOFTWARE ENGINEERING

Example SE Data	Example Mining Algorithms	Example SE Tasks
Sequences: Execution/Static Traces, Co-Changes	Frequent item set/sequence/partial-order mining, Sequence Matching/Clustering/Classification	Programming, Maintenance, Bug Detection, Debugging
Graphs: Dynamic/Static Call Graphs, Program dependence graphs	Frequent sub graph mining, Graph matching/clustering/ classification	Bug detection, debugging
Text: Bug Reports, e-mails, code comments, documentation	Text matching/clustering/classification	Maintenance, Bug detection, debugging

V. SOFTWARE ESCALATION PREDICTION WITH DATA MINING

It is a well appreciated fact in Software Engineering that defects reported in the software product post-delivery can severely impinge on its quality. In this context, it would be extremely useful if such defects can be predicted apriori. Data Mining can assist in such a prediction thereby leading to improved quality.

In the words of Bruckhaus et al., (2005) an escalation is triggered when a defect significantly impacts a customer's operations. Given the schedule constraints, it is always not easy to prioritize reported defects for resolution. The cost of escalations runs to millions of dollars and this sheds light for the need of Escalation Prediction (EP). The basic idea here is that if the vendor can predict escalation risks of known software risks, escalations can be prevented by fixing the high-risk defects before customers escalate them.

The main aspects of the solution are described.

1. Weekly data recollected from the vendor's Online Transaction Processing System (OLTP) and stored in a data mart.
2. The historical data collected in Step 1 is augmented by SPS Clementine data mining tool. The derived fields, historical information and available statistics are added to the set of available input fields. These include fields coming directly from the defect tracking systems, fields capturing bug histories, and escalation statistics for previous time periods.
3. Unsupervised clustering techniques are used to cluster the data, and the resultant data are available
4. The next step is training and validation of predictive models. The training involves

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

selection of a subset from among the 200 different input variables and setting values of training parameters for neural network in the case of clementine tool.

5. Once one or more satisfactory models are found, the most appropriate model is selected and is run against the most recent snapshot of the input data. The predicted escalations are reported to the product evaluation group.

To ensure the validity of the developed model, Bruckhaus et al., (2005) apply the technique to a live product development process and present the results that provide evidence that escalations can be indeed predicted.

Bruckhaus et al.,(2005) have attempted to use predictive technologies adopted in financial services industry to predict defects that have a high risk of escalation and thereby developers can deal with the "vital few" defects rather than the "trivial many". Such an application can result in significant improvement in Software Quality.

VI. Automated Software Testing Using Data Mining

The utility of software testing in software quality enhancement has been widely appreciated by the Software Engineering community. If such testing can be automated, they can deliver even more promising results in the endeavor to improve software quality. But there are few challenges in this regard. Sharma and Sharma (2011) suggest the following as challenges encountered in automated testing.

- Selection of a test tool
- Customization of the tool
- Selection of the automation level
- Development and verification of script
- Implementation of test management system

Sharma and Sharma (2011) present a novel approach to software testing that utilizes the amalgamation of automated software testing and data mining.

The approach of Sharma and Sharma (2011) for automated generation of test cases is diagrammed in Figure 1.

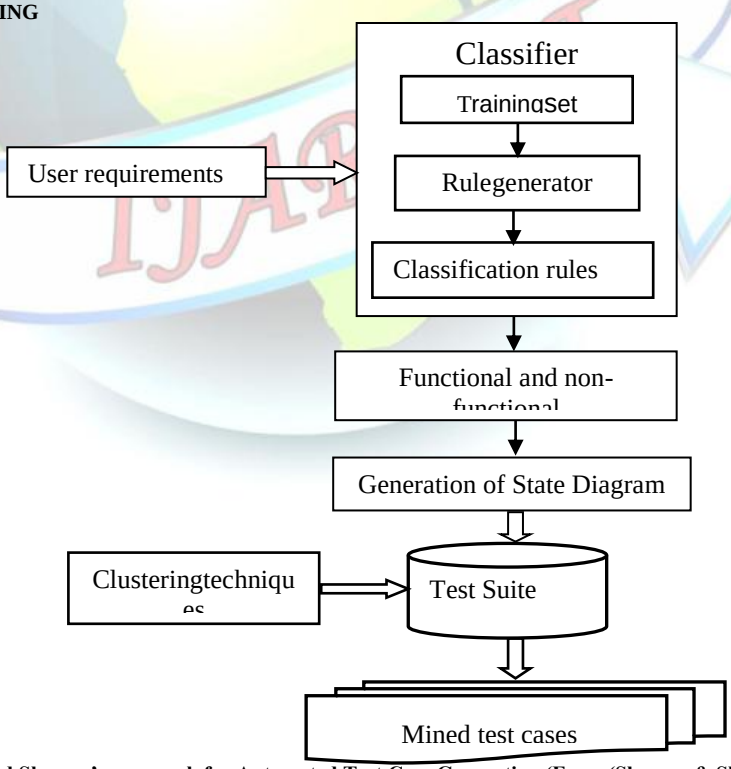

Figure 1 – Sharma and Sharma's approach for Automated Test Case Generation (From (Sharma & Sharma, 2011))

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

\

The major steps in this approach are:

1. A formal transformation of a detailed SRS (Software Requirement Specification) to a UML (Unified Markup Language) State Model
2. Generation of test cases from the state model
3. Mining of test cases

Sharma and Sharma (2011) also envision the introduction of agents in the presented approach to bring enhancement to the approach. Such an approach has the potential to greatly aid in automatic test case generation which is crucial for software quality enhancement.

VII. Data Mining For Management Of Software Development Process

The quality of the software is dependent on the process used for its engineering. Hence, it is vital to contemplate improvements in the management of the software development process in any endeavor to improve the quality of the software engineered. Alvarez et al., (2004) present a work on applying data mining to the management of software development process. They state that while the application of data mining to software engineering has an added problem 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. They use two tools – GAR which is based on unsupervised learning and ELLIPSES (Alvarez et al., 2002) which is based on supervised learning.

A. GAR

GAR is an algorithm that aids in the discovery of association rules on numeric attributes. It uses an evolutionary process for finding the most suitable intervals for each attribute that constitutes the rule.

In GAR, the model of a rule is described as:

$$if\ X_a \in [l_{ai}, l_{as}] \text{and } X_b \in [l_{bi}, l_{bs}] \text{and} \ldots \text{and } X_z \in [l_{zi}, l_{zs}]$$
$$\text{then } X_k \in [l_{ki}, l_{ks}] \text{and} \ldots \text{and } X_j \in [l_{ji}, l_{js}]$$

The fitness of a rule is based on several factors:

- Support – Rewards rules with a high value of support.
- Confidence – To orient the search process to find rules with high value of confidence.
- Before covered cases (recov) – To make the algorithm find novel rules in later searches.
- Number of attributes (natrib) – Rewards the number of attributes in a rule as rules with a high number of attributes tend to be of better quality.
- Amplitude (ampl) – Penalize individuals with very wide intervals

Fitness is obtained as

$$Fitness = (\sup port * w_s) + (conf * w_c) - (recov * w_r) + (natrib * w_n) - (ampl * w_a)$$

B. ELLIPSES

ELLIPSES is an evolutionary approach for classification that induces elliptical space regions. The regions are described by:

$$\frac{(x_1 - c_1)^2}{a_1^2} + \frac{(x_2 - c_2)^2}{a_2^2} + \ldots + \frac{(x_d - c_d)^2}{a_d^2} \le 1 \quad (2.2)$$

The fitness is calculated as

$$Fitness = clasif - nclasif - recov * frc + vol$$

where vol is the volume of the region, clasif is the number of instances of the majority class, nclasif is the number of instances of the other class, and recov is the number of instances covered by the other rules. The last term (frc) is weighted up by the expert depending on necessities. Values of frc close to 1 indicate a lower overlapping.

Alvarez et al.,(2002) state that the recent spurt in the tools for project management has led to the appearance of dynamic models for Software Development Projects (SDP). These tools are called "Software Project Simulators" (SPS). These dynamic models for SDP include a set of attributes related to the project environment, development organization, and the maturity level of the organization. For example, the group "project environment" may comprise attributes related to the initial estimations of the project; the group "Organization environment" may include the attributes related to management policies; the subgroup "Maturity degree" may include attributes bound to the average time of realization of certain activities of the project.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Issue 2, 2017 February

After defining the attributes, the manager decides the variables which are the subject of analysis. Examples include delivery time, cost, number of corrected errors and average productivity. The policies found by the mining process will be those that relate attribute values with variables. A value assigned to an attribute stays permanent. For attributes involving a degree of uncertainty, the manager may assign a range of values. The simulation tool generates a random value in the chosen range for each of the attributes, and a tuple of variables resulting from the simulation are produced. A record is generated for the values of the attributes and the values obtained for the variables of the project. The process is repeated a certain number of times, and the training file that serves as a base for the mining process is obtained.

Alvarez et al., (2002) suggest some adaptations to the GAR and ELLIPSES algorithms to make them amenable to the task at hand. To illustrate the validity of this method, they also present the results of simulating an already completed project and analyzing the data generated with the modified GAR and ELLIPSES.

In summary, Alvarez et al., (2002) exploit the conjunction of two techniques – dynamic systems simulation and data mining to obtain sound management rules that provide useful information about the characteristics of the project and ease decision making.

VIII. Conclusions And Future Work

Data Mining has immense scope in addressing the challenge of software quality enhancement. Some of the works presented in the paper are attempts to tap into the potential of data mining to address software engineering challenges. There have been attempts to improve the product as well as the software engineering process using data mining. Other techniques in Data Mining can be combined with recent advancements in computer science to uncover solutions to hitherto unaddressed and complex problems encountered in the engineering of quality software.

References

[1]. Alvarez, J.L., Mata, J., Riquelme, J.C, "Data Mining for the Management of Software Development Process", International Journal of Software Engineering and Knowledge Engineering, Vol.14, No. 6, pp. 665-695, 2004.

[2]. Alvarez, J.L., Mata, J., Riquelme, J.C. "Mining Interesting Regions using an Evolutionary Algorithm", 17th ACM Symposium on Applied Computing, pp. 495-502, 2002.

[3]. Bruckhaus, T., Ling, T., Madhavji, H., and Sheng, S., "Predicting Software Escalations with Maximum ROI", Proceedings of the 5th IEEE International Conference on Data Mining, pp. 717-720, 2005.

[4]. Gegick, M., Rotella, P., and Xie, T., "Identifying Security Bug Reports via Text Mining: An Industrial Case Study", 7th IEEE Working Conference Mining Software Repositories (MSR), pp. 11-20, 2010.

[5]. Hassan, A.E.,"The Road Ahead for Mining Software Repositories", IEEE Frontiers of Software Maintenance, 2008

[6]. Hayes, J.H., Dekhtyar, A., and Sundaram, S., "Text Mining for Software Engineering : How Analyst Feedback Impacts Final Results", 2005.

[7]. Kaufmann Han,J., and Kamber,M., "Data Mining: Concepts and Techniques", Second Edition, Morgan Publishers, 2006.

[8]. Sharma,S., and Sharma,A., "Amalgamation of Automated Testing and Data Mining: A Novel Approach in Software Testing", International Journal of Computer Science Issues, IJCSI, Vol. 8, Issue 5, No.2, 2011.

[9]. Wahidah,H., Low,P.V., Ng,L.K., and Ong,Z.L., "Application of Data Mining Techniques for Improving Software Engineering", School of Computer Sciences: Publications, 2011.

[10]. Xie, Thummalapenta, Lo and Liu, "Data Mining for Software Engineering", IEEE, Vol. 42, No. 8, pp. 55-62, 2009.

[11]. Xie,T., Thummalapenta, Suresh, Lo, David, Liu, Chao, "Data Mining for Software Engineering", International Journal of Computer Science Engineering and Information Technology (IJCSEIT), Vol.2, No.3, 2012.