SE code optimization using Data Mining Approach

S.Suyambu Kesavan
Research Scholar
Sivanthi Aditanar College
PillayarPuram, Nagercoil

Dr.K.Alagarsamy,
Associate Professor
Dept of MCA, Computer Center,
Madurai Kamaraj University, Madurai

Abstract:

Data mining also holds promises for other software engineering processes, which have to deal with uncertainty and intangible data such as cost estimation, effort estimation and quality. It can also aid in interesting by helping the software engineer to arrive at the require test cases that ensure accurate testing. Data mining also has potential to address some highly challenging areas of software engineering such as adaptability and security. This paper aims to help the software developer in the aspect of code debugging. Code debugging is an eminent phase of the software development. This paper gives some solution to rectify the code debugging problem using data mining technique.

Keywords: code debugging; Pattern matching, Software development,

Introduction:

Software quality is a big hectic in industry. Almost any software displays atleast some minor bugs after being released. Such bugs incur significant costs.A class of bugs which is particularly hard to handle is called crash. Occasional bugs that are failures which lead to faulty results with some but not with any input data. Noncrashing bugs in general are already hard to end. This is because nostack trace of the failure is available. With occasional bugs, the situation is evenmore difficult, as they are harder to reproduce. Developers usually try to findand rectify bugs by doing an in depth code review along with testing methods and classicaldebugging strategies. Since such reviews are very expensive, there is a need for tools whichlocalise pieces of code that are more likely to contain a bug.

The data mining algorithms works on software engineering data like text, sequences, graphs, which improves software engineering tasks like Programming, Maintenance, Bug Detection and Debugging. Implementation of source code management tool is done and finally data mining tools are implemented for Debugging Open Web API Mining.

Software engineers can start with either a problem driven approach or in practice they commonly adopt a mixture of the first two steps collecting and investigating data to mine and determining the SE tasks to assist. The three remaining steps are in order, preprocessing data, adopting, adapting and developing a mining algorithm and post processing applying mining results.
Processing data involved first extracting relevant data from the raw SE data for example, static method call sequences or call graphs from source code, dynamic method call sequences or call graphs from execution traces or word sequences from bug report summaries. This data is further processed by cleaning and properly formatting it for the mining algorithm. Like the input format for sequence data can be a sequence database where each sequence is a series of events.

The next step produces a mining algorithm and its supporting tool, based on the mining requirements derived in the first two steps. In general, mining algorithms fall into four main categories:

Data Clustering – Group the same data into clusters
Pattern Matching - Finding data instances for given patterns or matching pattern.

Frequent Pattern Mining - Finding most commonly occurring patterns.
Data Classification - Predicting labels of data based on the already labeled data.

The final step transforms the mining algorithm results into an appropriate format required to assist the SE task. For example, in the preprocessing step, a software engineer replaces each distinct method call with a unique symbol in the sequence data base being fed on to the mining algorithm. The mining algorithm then characterizes a frequent pattern with these symbols. In post processing, the engineer changes each symbol back to the corresponding method call. When applying frequent pattern mining, this step also includes finding locations that match a mined pattern for example, to assist in programming or maintenance and finding locations that violate a mined pattern for example, to assist in bug detection.

Discussion:

The objective of the research work to propose strategic Data Mining tools for program source code debugging which improves Software Reliability & Quality. We can implement Neglected conditions are an important but difficult-to-find class of software defects. This approach presents a novel approach for revealing neglected conditions that integrates static program analysis and advanced data mining techniques to discover implicit conditional rules using the novel approach for revealing neglected conditions that integrates static program analysis

Some of these legacy applications may have been in continuous use for more than 50 years like bank application. Such case the software industry is lax in keeping requirements and design documents up to date, so for a majority of legacy applications, there is no simple way to find out what requirements need to be transferred to the new replacement. However, some automated tools can examine the source code of legacy applications and extract latent requirements embedded in the code. These hidden requirements can be assembled for use in the replacement application. They can also be used to calculate the size of the legacy application in terms of function points, and thereby can assist in estimating the new replacement application. Latent requirements can also be extracted manually using formal code inspections, but this is much slower than automated data mining.

Most Software engineering data mining studies rely on well known, publicly available tools such as association rule mining and clustering. Such black box reuse of mining tools may compromise the requirements unique to software engineering by fitting them to the tools undesirable features. Further, many such tools are general purpose and should be adapted to assist the particular task at hand. However, Software engineering researchers may lack the expertise to adapt or develop mining algorithms or tools, while data mining researchers may lack the background to understand mining requirements in the software engineering domain. On promise way to reduce this gap is to foster close collaborations between the software engineering community and data mining Community.

This research effort represents one such instance. Writing Requirements is a two way process, classified as Functional Requirements and Non-Functional Requirements statements from Software Requirements Specification documents. This is systematically transformed into state charts considering all relevant information. The test cases can be used for automated or manual software testing on system level. A method is reduction of test suite by using mining methods there by facilitating the mining and knowledge extraction from test cases.

Proposed Method:

So far we have discussed about the problem occurs in source code management and various causes. In our research we are going to give an effective method to solve the source code management problems. This is a novel combination approach to join Software engineering with Data Mining. Since last few years this collaboration approach has been started with various directions for various problems. As the initiation process we have given the automated code correction tool for implementation part in software engineering.

Compiler or translator modules receive source code as input, break the code down into tokens and then output them in a new format. This output depends on the specific task performed by the module. The utility is based on three class groups: tokens, scanners and parsers. A scanner reads the code and breaks it down into tokens and returns them back to the parser. It also identifies the type of token to return. The parser requests successive tokens from the scanner and takes appropriate action before requesting the next token. The action of parser is to write out the token. These modules are generic enough too many programming languages with little modifications. Translator or compiler is just like sequence of programs nothing big in that but complexity of coding will much difficult compare with simple programming. Compiler checks the program and gives the error report or final result if not any error. In case errors occur in your program result won't come proper manner which gives lot of hectic to the developer stage in software engineering. Why don't we provide automatic error correction tool for to avoid these problems?

Here we have used combination of Hierarchical clustering algorithm and APRIORI algorithm. Based complexity program error will different but every language has predefined syntax and rules and regulation to type the code using that compiler trace the code line by line. Herewith I have given some sample machine language tokens. Compiler produces the machine language or computer understanding object. Using this object codes only program will get execute.

Class Name	**Class Description**
CEOLCommentToken	End of line comment.
CNumericToken	Any Numeric value
CEOF	Token End of file.
CEOL	Token End of line.

Suppose if any problem happen in the object code won't execute properly. To avoid these problems we are using online solution to overcome the existing problem. The code which contains the problem will take as the input for the searching after that we will apply the text mining technique. Text mining process contains some basic step we will discuss that.

- Information Extraction
- Information Gathering
- Information Processing:
  - Generate the unique words
  - Punctuation removing
  - Preposition removing
  - Root identification
  - Identification of most interesting terms

The above process is not easy one. Every phase contain tedious problem and different functionality here I am giving only minimized information. Here we have also used stemming algorithms to punctuation removal Part. There are many problem faced to implement this technique. The finalised result will give to hierarchical clustering analysis then final result will come as a solution for errors. Developer has to predict the exact solution among the result. First we have tried to among the solution automatically system will select one solution based on the apriori algorithm which included into program and once again program will execute but there lies some problem like source code change , cost increase etc. because we couldn't assure the solution will match up to 100 percent. If system developer chooses the solution, they are aware of the problem. This will help to

quick software development and reduce the developer work effort and time. We can also incorporate the concern database to solve the problem or finding the existing solution, which solved by previously.

Conclusion:

Computer invented for reducing the human efforts but trendy technology not like that, In our research we are giving the solution to software development industry to reduce the working effort, reduce problem complexity, code, time, money. Now IT field growing very fast and undertaking lot of projects. It's our belief this approach highly helpful to IT sector.

Future work:

In future we can also incorporate the concern database for searching. So that we can gets some good solution based on the company environment and product because most of the valuable coding and solution won't available in the website or internet. If we modified like this we can get some better solution in this.

Reference:

[1]. Tao Xie, Jain Pei, Ahmed E Hassen, "Mining Software Engineering Data", IEEE 29 th International Conference on Software Engineering
ICSE 07.

[2]. Ivano Malavelta, Henry Muccini, Patrizio Pellicciona, Damien Andrew Tamburri, "Providing Architectural Languages and Tools Interoperability through Model Transformation Technologies", IEEE Transactions on Software Engineering, Vol. 36, No. 4, January/February
2010, pp. 119-140.

[3]. Tao Xie, Suresh Thummalapenta, David lo, Chao Liu,"Data Mining for Software Engineering", IEEE Computer, August 2009, pp. 55-62.

[2]. Hamid Abdul BAsit, Stan Jarzabek, " A Data Mining approach for detecting higher-level clones in Software", IEEE Transactions on
Software Engineering, Vol. 35, No. 4, July/August 2009, pp. 497-514.

[4]. Mohammed J Zaki, Christopher D Carothes, Boleslan K Szymaski, "VOGUE: A Variable Order hidden Markov Model with duration based
on Frequent Sequence Mining", ACM Transactions on Knowledge Discovery from Data, Vol. 4 No.1, Article 5, January 2010.

[5]. Francisco P.Romero, Jose A.Olivas, MArcele Genero, Mario Piattini, "Automatic Extraction of the main terminology used in Empirical
Software Engineering through Text Mining Techniques" ACM ESEM 08 pp. 357 – 358.