Genetic k-means Clustering for Software Quality Estimation

S. Suyambu Kesavan, K. Alagarsamy, S. Palanikumar

Abstract – Software Quality Estimation has been a long standing and pressing problem for Software developers and managers for a period of time. In the current competitive business environment the paucity of resources prohibits managers from devoting resources to all modules to ensure quality. There have been attempts to use fault-data from previous system releases and construct fault prediction models. Such models are then used to predict the fault-proneness of modules in development. Modules that are predicted to be fault-prone are allocated more resources and subject to greater scrutiny and quality assurance techniques. The present paper proposes the use of genetic k-means clustering for software quality estimation.

I. INTRODUCTION

Clustering is a division of data into groups of similar objects. Each group called a cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups [1]. These clusters correspond to hidden patterns, and the search for clusters is termed “unsupervised learning”. One of the most popular clustering algorithms is the k-means clustering algorithm. Mertik et al. presented the use of advanced tool for data mining called Multimethod on the case of building software fault prediction model [5]. Azar et al. given a search-based software engineering approach to improve the prediction accuracy of software quality estimation models by adapting them to new unseen software products [6]. Prakriti and Rajeev presented set of software matrix that will check the interconnection between the software component and the application [7]. Naeem and Taghi presented a semi-supervised learning scheme as a solution to software defect modeling when there is limited prior knowledge of software quality [8]. Deepak et al. have studied three object oriented metrics and given a case study to show, how these metrics are useful in determining the quality of any software designed by using object oriented paradigm [9].

1.1 k-means Clustering

k-means clustering algorithm follows a simple way to classify a given data set as a certain number of clusters fixed a priori. The algorithm starts by defining k-centroids, one for each cluster. The better choice to place the centroids is to place them as far as possible from each point. The algorithm then proceeds to take each point in the data set and associate it with the nearest centroid. When all points are done this way, the first iteration is completed and an early groupage is done. Now the algorithm recalculates k new centroids. After this a new binding has to be done between the same set of data points and the new centroids. The k-centroids change step by step until no more changes are done. The algorithm aims at minimizing an objective function which is the squared error function. The objective function

\[ j = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_{ij} - c_j||^2 \]

where \( ||x_{ij} - c_j||^2 \) is a chosen distance measure between a data point \( x_{ij} \) and the cluster centre \( c_j \), is an indicator of the distance of the n data points from their respective cluster centres.

1.2 Genetic k-means clustering

Krishna and Murty propose a novel hybrid genetic algorithm that finds a globally optimal partition of a given data into specified number of clusters [2]. They attempt to hybridize GA with the k-means algorithm. The important aspects of the proposed GKA are listed below:

- Coding – W is encoded into a string \( s_w \) by considering a chromosome of length \( n \) and allowing each allele to take values \( [1, 2, \ldots, K] \). Each allele represents a pattern and the allele value indicates the cluster number to which the pattern belongs. This is called string-of-group-numbers encoding.

- Initialization – as with most GA’s the initial population is obtained by initializing each allele in the population to a random number selected from the set \( [1, 2, \ldots, K] \).

- Selection – a chromosome is selected from the previous population according to the distribution:

\[ P(s_j) = \frac{F(s_j)}{\sum_{j=1}^{N} F(s_j)} \]

where \( F(s_j) \) represents the fitness value.

- Fitness function – in order to minimize \( S(W) \) – the Total Within Cluster Variation, Krishna and Murty resort to the \( \sigma \) -truncation mechanism. They define

\[ f(s_w) = -S(W), g(s_w) = f(s_w) - (\chi - c)\sigma \]

where \( \chi \) and \( \sigma \) denote the average value and
Module Dependency Graphs are often used to represent the structure of complex systems. Modules are represented as nodes and their relationships are represented as edges. The complexity of the MDG of a modest sized system can itself be quite daunting and therefore, finding good partitions of the MDG where each partition can be relatively simpler compared to the entire system can be extremely useful. The huge dimensionality imposed by MDG’s necessitates the use of a Genetic Algorithm to perform clustering.


2.2 Software Quality Estimation with Clustering

Zhong, Khoshgoftaar and Seliya claim that Software Quality Estimation using supervised techniques like classification works fine only if past data from similar software projects are available [4]. They attempt to use cluster analysis with expert input for predicting the fault-proneness of software modules.

The basic idea behind the work of Zhong et al. is as follows: In the absence of software proneness labels, which might result from an organization dealing with a particular type of project it has not dealt with before, Unsupervised learning method like cluster analysis represent a viable option. Software modules are grouped into clusters based on the values of various metrics. Modules that are fault-prone will have similar values of metrics and thus be grouped in one cluster. Modules that are not fault-prone will constitute another cluster. After the cluster analysis is complete, an expert can inspect each cluster and label it as “fault-prone” or “not-fault-prone”. Exercising this option saves a lot of effort for the expert who would otherwise have to analyze every module to determine it’s fault-proneness.

For a module under study, the various metrics associated with it can provide indications of various aspects of it. For example, the McCabe’s Cyclomatic Complexity provides insight into the complexity associated with the module. This complexity may have implications for
understandability and maintainability. For module clustering, the values of these metrics form the basic input based upon which it is placed in a particular cluster.

III. PROPOSED METHODOLOGY

The basic methodology is very similar to the one adopted by Zhong et.al. The main objective was to determine whether the genetic k-means algorithm can deliver more accurate results than the regular k-means algorithm. Zhong et.al. also study the Neural Gas clustering algorithm but this research does not study that. A small local software organization involved in providing software solution to its vendors was subject to the study. The organization is responsible for the development and maintenance of web sites for it’s clients. Data pertaining to three web sites developed by the organization are studied. The Software Measurement and fault data pertaining to the 3 applications were collected at the module level – a module in this case was a method as all the web sites were developed using Java Server Pages (JSP). A total of 312 modules from the first application (called A1), 271 modules from the second (called A2) and 134 modules from the third application (called A3) were studied.

A set of nine metrics was used for each module: Total Lines of Code, Total Operators, Total Operands, Unique Operators, Unique Operands, Cyclomatic Complexity, Halstead’s Program Length, Halstead’s Program Volume, Branch Count

The nine metrics taken into account for the study do not in any way fully characterize a module. Many metrics are available and some of them may be even more accurate predictors of the attribute they are attempting to measure. The study concentrates on the values of only these modules. The choice of these metrics was primarily based on the interactions with the expert involved in labeling. The expert was the senior project manager with 15 years of experience in web site development, maintenance and management.

Both the k-means and the genetic k-means clustering algorithms were implemented in VB.NET under the Windows XP Operating System. The number of clusters to be generated heavily depends on the availability of resources. For the experiment, the number of clusters was fixed at 30, 20 and 15 clusters for the three applications A1, A2 and A3. The choice of the number of clusters is a trade-off between the effort required by the expert for labeling the clusters and obtaining a fine representation of the software data. The more the number of clusters, more is the effort required by the expert and finer the granularity.

For labeling, the expert is provided with the global mean, minimum, maximum, median, 75th percentile, 80th percentile, 85th percentile and 90th percentile of each metric for each cluster as well as the size of each cluster. Based on these statistics, the expert labeled each cluster as “fault prone” or “not fault prone”. This is the same methodology followed by Zhong et.al.

For the comparative analysis, the following measurements are used. These are essentially the same ones used by Zhong et.al.

- Mean Squared Error (MSE)
- Cluster Purity – percentage of the most dominated category (fault-prone or not-fault-prone) in the cluster
- Average Purity – Mean purity of all the clusters

The expert’s labeling decision is evaluated using the following criteria:

- Over-all classification error
- False Positive rate (FPR) – percentage of not fault prone modules mislabeled as fault-prone
- False Negative Rate (FNR) – percentage of fault-prone modules mislabeled as not-fault-prone.

For the experiment, a module was considered fault prone if it had more than 5 faults. This was used only to assess the labeling performance of the expert and not for clustering.

IV. RESULTS AND DISCUSSION

The results of the experiment are tabulated below:

Table 1 - Clustering Quality Results for Application 1(A1)

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>1567.81</td>
<td>1395.32</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.797</td>
<td>0.877</td>
</tr>
</tbody>
</table>

Table 2 - Clustering Quality Results for Application 2(A2)

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>1317.78</td>
<td>1213.45</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.832</td>
<td>0.882</td>
</tr>
</tbody>
</table>

Table 3 - Clustering Quality Results for Application 3(A3)

<table>
<thead>
<tr>
<th>Measure</th>
<th>k-means</th>
<th>Genetic k-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Squared Error</td>
<td>877.78</td>
<td>801.21</td>
</tr>
<tr>
<td>Average Purity</td>
<td>0.871</td>
<td>0.893</td>
</tr>
</tbody>
</table>
Table 4 – Expert Labeling Performance with Application 1 (A1)

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>15.6%</td>
<td>67.2%</td>
<td>21.6%</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>15.8%</td>
<td>60.1%</td>
<td>20.9%</td>
</tr>
</tbody>
</table>

Table 5 – Expert Labeling Performance with Application 2 (A2)

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>16.2%</td>
<td>58.1%</td>
<td>19.8%</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>15.2%</td>
<td>53.5%</td>
<td>18.6%</td>
</tr>
</tbody>
</table>

Table 6 – Expert Labeling Performance with Application 3 (A3)

<table>
<thead>
<tr>
<th></th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-means</td>
<td>13.6%</td>
<td>39.2%</td>
<td>19.8%</td>
</tr>
<tr>
<td>Genetic k-means</td>
<td>12.9%</td>
<td>35.7%</td>
<td>18.6%</td>
</tr>
</tbody>
</table>

A closer analysis of the results reveals certain interesting properties of the Genetic k-means algorithm. First, considering the clustering quality, it is observed that Genetic k-means tends to produce clusters with a lower mean squared error value compared to k-means. The difference tends to be more pronounced for the larger application which is A1 and the difference tends to narrow down for the smallest application A3. This leads to the conclusion that Genetic k-means has more...
potential to perform when the number of modules to be clustered is huge as is the case with many real-world projects. The same conclusion holds for the average purity as well. Genetic k-means produces clusters with more average purity and the difference increases with the increasing size of the project considered. This stands in line with the general principle that as the search space tends to be come huge, genetic algorithms have more potential to deliver compared with conventional ones.

When analyzing the expert labeling decision, the first thing to be noted is that the feedback from the expert indicated that clusters generated by genetic k-means were more easier to label that those generated by k-means. This tends to be because of the ability of the genetic k-means algorithm to produce more coherent clusters. Narrowing the analysis to false-negative misclassification rates, as the cost of such misclassifications is significantly higher than that of false-positive misclassifications, the genetic k-means is found to perform better with the increasing size of the applications considered. It is very important to consider false-negative misclassifications because if a faulty module is classified as non-faulty, the impact will be severe. On the other hand if a not faulty module is classified as faulty the only negative outcome will be the increased wasted resources spent on them.

V. CONCLUSION

The potential of the genetic k-means algorithm to cluster software modules based on their fault proneness was studied. It was found that the genetic k-means algorithm performs significantly better than k-means as the size of the application increases. Given the paucity of resources in most organizations such ability to predict the fault proneness of modules under development is vital. Clustering has the added advantage that it can perform even in the absence of fault and measurement data of similar past software projects. In this regard, the results of the conducted empirical study demonstrate that as the size of the application increases genetic k-means can prove to be more useful and accurate than k-means clustering.

REFERENCES


ABOUT THE AUTHORS

S. Suyambu Kesavan M.C.A, M.Phil., He has 10 years teaching experience. His area of interest is in the field of Software engineering, Data mining and Soft computing. He is a research scholar at manonmaniam
sundaranar university and working as a Programmer, Dept. of Computer Science, Sivanthi Aditanar College, Nagercoil. (email:nellaikesavan@gmail.com)

K.Alagarsamy Ph.D. He is a Associate professor, MCA Dept., School of Information Technology, Mudurai Kamaraj University, Mudurai. He has 32 years of teaching service. His area of interest in the field of Software Engineering, Data mining and Soft computing. He has published more than 25 papers in various national and international journals. (email: alagarsamymku@gmail.com)

S.Palanikumar B.E., M.E.,Ph.D., He has 1.5 year of Industry experience and 11.5 years of teaching experience. His area of interest are in the field of Image and Signal Processing, Software Engineering and Bio-metrics and Embedded and Computer communication Systems. He received his doctoral degree from Anna University, Chennai. And working as an Associate Professor, IT Dept. at Noorul Islam Centre for Higher Education, Noorul Islam University, Kumaracoil. He has published more than 18 papers in national and international journals. (email:palanikumarcse@yahoo.com)
A Genetic Algorithm based Association Rule Mining Approach for Identifying Important Human Factors that Impact Software Quality

S. Suyambu Kesavan, K. Alagarsamy, S. Palanikumar

Abstract—Over the years, the production of quality software has become a big challenge to software organizations, developers and managers. While addressing the issue of Software Quality, the role played by the human factors seems to be the most ignored of all. Human factors engineering pertains to the application of scientific knowledge concerning human behavior to the design of jobs and products. The paper presents a genetic algorithmic approach for the discovery of association rules pertaining to human factors in the engineering of quality software.

I. INTRODUCTION

Over the years, the production of quality software has become a big challenge to software organizations, developers and managers. Software Engineers and managers grapple with a lot of challenges in their endeavor to engineer quality Software. The American Heritage dictionary defines quality as “an attribute or characteristic of something” [1]. Often these attributes are measurable – like cost, color, length. But given the nature of software, it is difficult to characterize software through measurable properties like cyclomatic complexity and lines of code do exist. According to Pressman, there are 2 kinds of Quality:

- Quality of Design – characteristics that designers specify for an item.
- Quality of Conformance – the degree to which design specifications are followed during manufacturing.

But Software Quality is not that easy. In DeMarco ‘s view a product’s quality is a function of how much it changes the world for the better [2]. Pressman defines Quality as “conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software”[1].

Pressman categorizes the factors that affect software quality into two main groups – those that can be measured directly and those that can be measured only indirectly [1]. Examples of factors that can be directly measured include defects per function point. Examples of factors that can be measured only indirectly are usability and maintainability.

II. HUMAN FACTORS

While addressing the issue of Software Quality, the role played by the human factors seems to be the most ignored of all. Human factors engineering pertains to the application of scientific knowledge concerning human behavior to the design of jobs and products. The dearth of available literature pertaining to the role of human factors in Software Quality stands testimony to this fact. While a lot of works identify the human factors involved in software quality and productivity, little has been done in the application of scientific methodologies to draw meaningful and valid conclusions pertaining to human factors. McConnel goes to the extent of stating that people issues have the biggest impact on software quality [3].

While addresses the people factors that have an impact on productivity of software [4]. White suggests that it is important to have the right people, people who are knowledgeable, skilled and satisfied, project manager who works well with people to ensure quality.

Fairly states 17 different quality and productivity factors [5]. Out of the 17 factors, the following are those pertaining to people:

- Individual Ability – encompasses both the general competency of the individual and familiarity with the particular application area
- Team Communication – the ability of programmers to communicate well have a direct bearing on the quality
- Required Skills – Software Engineering requires a variety of skills
- Adequacy of Training – the degree to which the training requirements of personnel are met
- Management skills – the differences in the nature of software projects compared with others dictate the need for new skills for managing software projects

John, Maurer and Tessem opine that since software is developed for people and by people, human and social factors have a very strong impact on the success of software development endeavors and the resulting
system [6]. They reinforce the opinion expressed above that surprisingly, much of the software engineering research in the last decade has been technical ignoring the people aspect. Grisham and Perry attempt to derive the conditions under which extreme programming – an agile method – can lead to user satisfaction. They analyze whether on-site customer can always lead to better relationships.

**III. ASSOCIATION RULE MINING**

Data Mining is generally considered as the process of discovering hidden, nontrivial and previously unknown patterns from a large collection of data. Association rule mining is an important component of data mining. According to the Indian Agricultural Statistical Research Institute (IASRI), association rule mining is perhaps the most widely studied model by the data mining community [7]. Xiaowei et al. designed a genetic algorithm-based strategy for identifying association rules without specifying actual minimum support [10]. Ali et al. presented a new algorithm, Cluster-Based Multi-Objective Genetic Algorithm (CBMOGA) which optimizes the support counting phase by clustering the database [11]. Nath et al. described a comprehensive survey on the state-of-the-art algorithms for association rule mining, especially when the data sets used for rule mining are not static[12]. Bin and Yuefeng given a hybrid association rule mining method for characterizing network traffic behavior [13]. Mohammed and Bassam indicated the limitation of the original Apriori algorithm of wasting time for scanning the whole database searching on the frequent itemsets, and presented an improvement on Apriori by reducing that wasted time depending on scanning only some transactions [14]. Potential examples of the utility of the mined association rules abound in almost every discipline. Organizations, particularly, in the retail segment are interested to use association rule mining to discover customer buying patterns that can aid them in major decision making regarding which products can be promoted together. Medical domain is no exception. Association rules can help doctors find the factors most likely to cause a particular disease. This is especially significant for non-communicable diseases for which the importance of various contributing factors is relatively unknown. Association rule mining can also be applied to agricultural databases to survey data from agricultural research.

The basic objective of association rule mining is to find all co-occurrence relationships called associations. The classic application of association rule mining is the market basket analysis which aims to discover how items purchased by customers are associated. An association rule is of the form $X \rightarrow Y$ where $X$ and $Y$ are collections of attributes whose intersection is null. For example, every customer who purchased a computer ($X$) also purchased a printer($Y$). $X$ is called the antecedent and $Y$ is called the consequent. Since the number of possible association rules can be huge, often the interest is on those rules which satisfy some constraints. The most common of these constraints include support and confidence.

Formally, let $I = \{i_1, i_2, ..., i_m\}$ be a set of items. Let $T = \{t_1, t_2, ..., t_n\}$ be a set of transactions where each transaction $t_i$ is a set of items. An association rule is of the form $X \rightarrow Y$, where $X \subseteq I, Y \subseteq I$ and $X \cap Y = \emptyset$. $X$ (and $Y$) is a set of items called an itemset. The support of a rule $X \rightarrow Y$ is the percentage of transactions in $T$ that contains $X \cup Y$ and can be stated as $Pr(XUY)$ which is the estimate of probability. If $n$ is the number of transactions then the support of the rule $XUY$ is given as $(XUY).count/n$. The confidence of a rule $X \rightarrow Y$ is the percentage of transactions in $T$ that contain $X$ also contain $Y$. It can be stated as $Pr(Y|X)$ – the conditional probability.

Many algorithms exist for discovering Association Rules. Apriori and FP growth are popular representatives in this category.

**IV. GENETIC ALGORITHMS**

Genetic Algorithms belong to the class of evolutionary computation and attempt to mimic the natural process of evolution to uncover solutions to problems. The basic ideas behind them were presented in the previous chapter. The pseudo-code is sketched here for completeness.

1. Initialize the population by generating an initial set of chromosomes representing solutions
2. Calculate fitness of the individuals in the population based on some criteria
3. Select 2 individuals from the population at random based on the fitness so that the more fit individuals get more chance for selection
4. Perform cross over by selecting one or more cross points
5. Select chromosomes at random and Mutate them with some probability
6. If the terminating condition is met stop and display the best solution obtained. Else repeat steps 2 – 5 using the newly generated solutions

Genetic Algorithms have been extensively applied in various domains and are very promising in yielding solutions to complex problems involving a lot of uncertainty.
V. MINING ASSOCIATION RULES BETWEEN HUMAN FACTORS AND QUALITY OF SOFTWARE

The importance of human factors in assuring software quality was elaborated earlier. The research aims at deriving association rules that would aid in discovering the crucial human factors that have a significant impact on the quality of the software engineered.

The list of human factors involved in software engineering and those that impact the quality of the software tends to be very huge and most of the factors may also be correlated. For the purpose of the research the following set of human factors is considered. The list is by no means complete and can be extended to accommodate many more.

<table>
<thead>
<tr>
<th>Human Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competency in Technology</td>
<td>The skill in the particular technology</td>
</tr>
<tr>
<td>Experience in the technology</td>
<td>The experience of the candidate in the technology</td>
</tr>
<tr>
<td>Experience in the domain</td>
<td>The experience in the domain of the project</td>
</tr>
<tr>
<td>Ability to work in a team</td>
<td>The ability of the individual to contribute in a team</td>
</tr>
<tr>
<td>Communication Skills</td>
<td>The ability of the individual to comprehend requirements and communicate problems faced</td>
</tr>
<tr>
<td>Motivation Level</td>
<td>The level of motivation of the individual</td>
</tr>
<tr>
<td>Commitment</td>
<td>The commitment of the individual toward the success of the project</td>
</tr>
<tr>
<td>Responsibility</td>
<td>The willingness of the individual to learn from past mistakes</td>
</tr>
</tbody>
</table>

5.1 Methodology

A project manager working in a local software organization and who has 8 years of experience in software development and 7 years in software project management is asked to rate the team leaders of 50 teams that have delivered around 60 web sites. The team leaders are rated in the scale of 0 to 3 on all the above factors where 0 indicates that the attribute in question is least and 3 indicates that the attribute in question is maximum. For example, a rating of 0 for experience in the domain factor means the individual has no experience in the domain and a rating of 3 for motivation level means that the individual is motivated to the greatest extent. Despite the subjectivity involved in rating of the team leaders, the experience of the manager is expected to increase the objectivity to a reasonable level and thus improve the quality and utility of the mined association rules. The quality information pertaining to the 50 web sites is available. For simplicity, every web site is classified either as quality or non-quality web site based upon the number of issues faced post delivery for the website. Out of the 60 web sites, 43 are classified as “quality” web sites and the remaining 17 as “not-quality” sites. This categorization is done solely based on the number of issues reported so far in the web sites and by comparing this with a threshold value. A quality value of 1 indicates a quality web site and a 0 indicates a no-quality web site.

5.2 Applying GA to mine association rules

To apply Genetic Algorithm to any problem the first issue that needs to be addressed is encoding of the solutions. Solutions have to be represented as chromosomes. When applying Genetic Algorithm to mine association rules there are basically 2 approaches for solution representation. In the Pittsburgh approach where each chromosome represents a set of rules and the Michigan approach where each chromosome represents a single rule. The research uses a modified Michigan approach proposed by Ghosh and Nath [8]. The basic idea is to associate 2 bits with each attribute. If these 2 bits are 00, then the following attribute appears in the antecedent part and if these 11, then the following attribute appears in the consequent part. The remaining 2 combinations – 01 and 10 imply the absence of the following attribute in the rule. Since, the position of various attributes in the chromosomes is fixed and the attributes are numeric, the values of the attributes can be encoded in their binary form. The values to be encoded are in the range of 0-3 for the ratings and 0-1 for quality attribute. 8 human factors would need 2 bits each and the quality attribute needs 1 bit. For all the 9 attributes 2 tag bits are required as described above. The total space requirement for association rules of arbitrary length turns out to be 8 * 2 + 1 + 9 * 2 = 35 bits. The fitness evaluation is done using an approach analogous to the one adopted by Wakabi-Waiswa and Baryamureeba [9]. The Mutation probability was set at 0.01. The number of generations is capped at 150.

5.3 Results and Discussion

The best performing individuals in the 150 generations are sorted according to their fitness values and from the resulting 150 rules, rules for which the consequent is some attribute other than the quality attribute are pruned. The top 5 association rules that were mined are shown in the Table 2.
Table 2 - Top 5 Rules with Quality Attribute as the consequent (in terms of fitness)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Time</th>
<th>Average Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation Level=3 ^ Experience in the domain=3  →  Quality=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience in the domain =2 ^ Ability to work in team =3  →  Quality=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience in Technology = 3 ^ Experience in domain = 2 ^ Motivation = 1  →  Quality=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience in Technology = 0 ^ Ability to work in team = 1  →  Quality=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commitment = 3 ^ Responsibility = 3  →  Quality=1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The top 3 rules in terms of fitness imply that a high motivation and high domain experience are likely to yield good quality software. Interestingly, the competency in the technology attribute is not present in the top 5 rules even though it is generally perceived as highly important for quality. Another interesting rule that was mined as Motivation = 3  →  Commitment = 3 ^ Responsibility = 3. This rule, though, was not considered as it gives no indication of quality, reveals an interesting correlation that highly motivated people tend to be more responsible and committed. The third rule gives an insight into one important aspect of human factors pertaining to quality. Notwithstanding the experience in the domain and technology, less motivated individuals tend to impact the quality negatively. Experience seems to offer no guarantees of quality. 78.21% of the mined rules that had quality = 1 as the consequent had a motivation level of 2 and 3 in the antecedent. This is in line with the intuition that motivated individuals tend to overcome all other obstacles they might otherwise have pertaining to experience and competency.

5.4 Comparison with Apriori and FP Growth

To demonstrate the potential of GA, the same experiment was done with the Apriori Algorithm and the FP Growth algorithm. The following table gives a comparative statement about the performance of the algorithms in terms of execution time and average quality. In terms of time Apriori seemed to be the worst performer and this is expected given the time required for generating candidate item sets by the algorithm. Though the difference between GA and FP Growth in terms of time is very small, there is a considerable improvement in the quality of the rules generated. GA seems to generate more quality rules than FP Growth. This reinforces the opinion that as the complexity of the problem and uncertainty involved increases, trends shift in favor of GA.

Table 3 - Comparison of Apriori, FP Growth and GA

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (secs)</th>
<th>Average Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>2.09</td>
<td>78.91%</td>
</tr>
<tr>
<td>FP Growth</td>
<td>1.81</td>
<td>80.11%</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>1.78</td>
<td>84.12%</td>
</tr>
</tbody>
</table>

Figure 1 – Comparison in terms of time

Figure 2 – Comparison of Apriori and GA in terms of average quality
VI. CONCLUSION

A genetic algorithm was proposed to mine association rules to discover interesting associations between human factors and software quality. Human factors have a great impact on software quality and a careful identification of the significant human factors contributing to high quality is of utmost importance for project managers and organizations in their endeavor to develop quality software. Several association rules are mined and the mined association rules indicate that motivation is one of the crucial human factors impacting quality. Experience in domain, commitment and responsibility seem to be the other dominating factors in ensuring high quality. The research attempts to throw light on the prospects of utilizing Association rule mining and genetic algorithm in identifying the crucial human factors impacting quality. Given the strong correlation between human factors and quality this can be a starting step of an attempt that can lead to perceivable improvements in the quality of the software developed. Managers can reap great benefits by paying attention to the crucial human factors identified. In future, the list of human factors considered can be extended to include more and interesting associations discovered between them and software quality.

REFERENCES


ABOUT THE AUTHORS

S.Suyambu Kesavan M.C.A, M.Phil.. He has 10 years of teaching experience. His area of interest is in the field of Software engineering, Data mining and Soft computing. He is a research scholar at manonmaniam sundaranar university and working as a Programmer, Dept. of Computer Science, Sivanthi Aditanar College, Nagercoil.

K.Alagarsamy Ph.D. He is an Associate professor, MCA Dept., School of Information Technology, Mudurai Kamaraj University, Mudurai. He has 32 years of teaching experience. His area of interest in the field of Software Engineering, Data mining and Soft computing. He has published more than 25 papers in various national and international journals. (email: alagarsamymku@gmail.com)

S.Palanikumar B.E., M.E., Ph.D.. He has 1.5 year of Industry experience and 11.5 years of teaching experience. His areas of interest are in the field of Image and Signal Processing, Software Engineering and Bio-metrics and Embedded and Computer communication Systems. He received his doctoral degree from Anna University, Chennai. And working as an Associate Professor, IT Dept. at Noorul Islam Centre for Higher Education, Noorul Islam University, Kumburacoil. He has published more than 18 papers in national and international journals. (email: palanikumarasc@yahoo.com)
Software Reliability Classification using Genetic Algorithm
S. Suyambu Kesavan, K. Alagarsamy, S. Palanikumar

Abstract—The engineering of reliable software has become a top priority of the day in the zeal of software developers to develop quality software. It would be of great benefit to developers if they can have an idea of the reliability of the modules developed by them so that they can take corrective actions if required. The present paper proposes a reliability classification method for software modules based on certain metrics associated with the module. The approach uses genetic algorithm, for classification and the results of an empirical study conducted are greatly promising.

I. INTRODUCTION

Data mining is the process of identifying novel, potentially interesting and ultimately understandable patterns from large volumes of data [1]. Classification is an important concept in data mining. The classification problem in data mining can be stated as follows: given some attributes for data items and their associated class labels, predict the class label attribute for data items for which it is unknown. The potential applications of classification can be enormous. For example, classification may be used to discover the classes of customers who are likely to buy a product and customers who will not. Likewise, classification can be used to classify students as performers and under-performers so that increased attention may be given to the students likely to under-perform. Classification can also be used to uncover patterns likely to be found in fraudulent transactions. This information in turn can be useful to identify and block such fraudulent transactions in the future.

Several techniques and algorithms for classification can be found widely in the literature. The most popular ones are decision tree induction, Bayesian classification, Support Vector Machines (SVM)[2], Neural Networks, Rough Set Theory, k-nearest neighbor classifiers and Genetic Algorithms. This paper uses the genetic algorithmic approach for classification.

1.1 Genetic Algorithms

Genetic Algorithms belong to the class of evolutionary computation and are highly suited for searching and optimization problems. These genetic algorithms seek to find solutions to complex problems by emulating the natural process of evolution. Chapter 3 provided an insight into the working of Genetic algorithms.

To use Genetic Algorithms for classification, randomly generated rules form the initial population. As an example, the rule “If Blood Pressure > 160 and Blood Sugar > 300 Then Heart_Failure=yes”, can be encoded as 111. Similarly the rule “If Blood Pressure Not > 160 and Blood Sugar > 300 Then Heart_Failure=no is encoded as 010.

A new population consisting of the fittest rules in the current population is formed and the formation of this new population called offspring entails the application of operators like cross-over where substrings from a pair of rules are swapped to form new rules and mutation where selected bits are inverted. The fitness of a population is assessed using a fitness function that may be the classifier performance on the training samples. The process of creating new populations continues until a terminating criterion is met.

II. SOFTWARE RELIABILITY

Engineering of software that can be trusted to provide expected service to its users has been a long standing challenge confronting software practitioners. The basic idea is to ensure that the software delivered to the customer is reliable before its delivery. A lot of roadblocks deter the software engineer from achieving this. The main issue seems to be the lack of sound procedures that aid in measuring the reliability of the software. Since what cannot be measured cannot be controlled, engineering of reliable software is a pressing problem facing the software community. The next few paragraphs are devoted to the understanding of the concept of reliability.

Laprie defines “Dependability” as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers [3]. According to Sommerville, Reliability is one aspect of dependability [4]. The aspects of dependability as listed by Sommerville include:

- Availability – the probability that the system will be up and running and able to deliver useful services at any given time
- Reliability – the probability over a given period of time that the system will correctly deliver services as expected by the user
- Safety – the likelihood that the system will cause damage to people or its environment
- Security – the likelihood that the system can resist accidental or deliberate intrusion

This research focuses on Reliability. The IEEE Definition of Software Reliability is given as “the ability of a system or component to perform its required functions under stated conditions for a specified period of time”. According to Karanta [5], reliability estimation can follow one of 2 approaches:
Approach 1 – code analysis using static analyzers, model checkers and theorem provers. An elaborate discussion on this approach can be found in [6].

Approach 2 – analysis of software from an external point of view. Techniques include testing, expert opinion, and operational data.

Neither approach is completely sufficient by itself. Testing can only prove the presence of errors not their absence. These unnoticed errors often manifest themselves as faults which result in the system failure where the system fails to meet its expected requirements. Static code analyzers are limited by the knowledge of the developers of the analyzers and can find only those faults they are programmed to find. The field of development of static code analyzers and model checkers is in its infancy and this will definitely take time to evolve.

With this status quo, there stems an imperative necessity to explore alternative ways to estimate reliability. Many works have been reported in the literature in this direction out of which the work of Chitra, Madusudhanan and Rajaram forms the motivation of the research outlined in this paper [7].

III. RELATED WORK

Chitra, Madusudhanan and Rajaram present a neural network based approach for software reliability exploration. Their idea is to use a neural network based classifier to classify a software module as reliable or not. The main criterion used for arriving at such a conclusion is the cyclomatic complexity density. Employing a neural network with 1 hidden layer containing 25 nodes and 0.1 step size for gradient descent Chitra et. al. report a very low classification error rate of 0.38% on the KC3 data set. The work of Chitra et. al. throws light on the prospects of applying novel advances in computer science to the problem of software reliability exploration.

Bijamma et al. discussed about software reliability model in their work [10]. AnilPrincy and Sridhar given a comparative analysis of software reliability models [11]. Alessandra et al. described about cost effective reliability model[12]. Cotroneo et al. discussed about fault analysis in open source software in their work [13]. Alonso et al. described about fault repairs and mitigations in space missile system software [14],[15]. Wason et al. discussed about software reliability model estimation [16].

IV. SOFTWARE RELIABILITY CLASSIFICATION USING GENETIC ALGORITHM

The research attempts to capitalize on the power of genetic algorithms in uncovering accurate classification models. The problem can be stated as below – “Given a set of characteristics of a software module (metrics), classify it as reliable or unreliable”. Such identification can alert the software project manager to the need of extensive testing of the modules predicted to be “unreliable”. This in turn can reveal faults which can be corrected before the software is delivered.

4.1 Metrics considered for the study

A module is represented by a set of metrics. These metrics are measurements of certain properties of the module under consideration. Computation of these metrics values can be done manually or many automated tools are available for their computation.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALOC</td>
<td>Mean Lines of Code per Method</td>
</tr>
<tr>
<td>MLOC</td>
<td>Median Lines of Code per Method</td>
</tr>
<tr>
<td>ADEC</td>
<td>Mean number of decisions per Method</td>
</tr>
<tr>
<td>MDEC</td>
<td>Median number of Decisions per method</td>
</tr>
<tr>
<td>Cyclomatic Complexity</td>
<td>The Cyclomatic complexity of the module</td>
</tr>
<tr>
<td>Halstead’s Program Length</td>
<td>Halstead’s Length metric</td>
</tr>
<tr>
<td>Halstead’s Program Volume</td>
<td>Halstead’s Volume metric</td>
</tr>
</tbody>
</table>

4.2 Experimental Setup

In order to build a genetic algorithm based classifier, first 213 modules developed by a local software organization are used as training samples. The reliability classes of these modules are known. For the purposes of the experiment, a module with 5 or more reported number of faults in the first three months was considered as “unreliable” and that with less than 5 faults was considered “reliable”. Out of the 213 selected modules, 181 were in the reliable category and 32 were in the “unreliable” category. The values for all the metrics listed in Table 5.2 are known for the 213 modules.

In order to apply genetic algorithm for the problem, solutions are represented by chromosomes containing genes that are the values of the metrics. The fitness function is evaluated by using the Linear Discriminant Analysis (LDA) – which is a classifier strategy proposed by Duda [8] and used by Vivanco and Pizzi with the leave-one-out method of training and testing [9]. This means that first a module is selected and training is done with all the remaining 212 modules and it is observed whether the selected module is correctly classified. The process is repeated for all the 212 modules. The number of chromosomes in a population was set at 200 and the number of elite genes is fixed at 50. This means that in each generation after the chromosomes are sorted in decreasing order of fitness, the top 50 chromosomes are
passed on to the next generation as “elite” genes. A random probability $P$ is generated and from the remaining 150 genes, 2 genes with fitness greater or equal to $P$ are selected as parent genes. A cross-over point is selected at random and a new offspring is produced by combining bits from both the parents. If the generated probability is greater than $1 - 10\% = 0.9$, the offspring is mutated by changing each bit from 0 to 1 and vice versa. The process of cross-over and mutation are repeated until 150 new offsprings are created and with the new population of the 50 elite genes plus the 150 offsprings the whole process is repeated for 200 generations. The classification rate was the number of times the module left out was correctly grouped.

V. RESULTS AND DISCUSSION

The results obtained using the genetic classifier, when tested with the 213 modules are tabulated below.

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Actual Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELIABLE</td>
<td>167</td>
</tr>
<tr>
<td>UNRELIABLE</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 2 – Classification Confusion Matrix

Table 3 – Error Report

<table>
<thead>
<tr>
<th>Class</th>
<th># Cases</th>
<th># Errors</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELIABLE</td>
<td>181</td>
<td>14</td>
<td>7.73%</td>
</tr>
<tr>
<td>UNRELIABLE</td>
<td>32</td>
<td>2</td>
<td>6.25%</td>
</tr>
<tr>
<td>Overall</td>
<td>213</td>
<td>16</td>
<td>7.51%</td>
</tr>
</tbody>
</table>

As the results indicate, GA achieves very promising results of misclassification of 6.25% unreliable modules as reliable and 7.73% reliable modules as unreliable. It is important to note the significant aspect of the difference in misclassification of reliable and unreliable modules. A reliable module misclassified as unreliable does not cause much harm except that it might incur some wastage of resources. But a unreliable module misclassified as reliable might have disastrous consequences as faults in such modules can escape testing and reach the user. Yielding a smaller misclassification of unreliable modules as compared to reliable modules is a positive aspect of the developed GA based classifier.

The following charts are a graphical representation of the stated results.

In order to demonstrate the superiority of GA in solving such complex classification problems, the research attempted to perform the same classification with neural network multi-layer feed forward classification followed by Chitra et. al. The results obtained using these approaches are tabulated below:

<table>
<thead>
<tr>
<th>Class</th>
<th># Cases</th>
<th># Errors</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELIABLE</td>
<td>181</td>
<td>23</td>
<td>12.71%</td>
</tr>
<tr>
<td>UNRELIABLE</td>
<td>32</td>
<td>4</td>
<td>12.5%</td>
</tr>
<tr>
<td>Overall</td>
<td>213</td>
<td>27</td>
<td>12.68%</td>
</tr>
</tbody>
</table>

Table 4 – Error Report using Neural Network Feedforward Classification

As can be observed GA provides more accurate classifications compared to the Neural Network classification. The various parameters for the neural network are the same as followed by Chitra et. al [7].
VI. CONCLUSION AND FUTURE WORK

Software Reliability Exploration is fraught with difficulties and various approaches have been proposed for the same. These include the usage of static code analyzers and testing. Such approaches are insufficient when used in isolation and need to be complemented by other techniques.

In this direction, the research proposed a Genetic Algorithm based classifier for software reliability classification. The empirical results reveal that GA classifiers can yield fairly good classifications. The GA classifier developed reports a very low error rate of 6.25% for unreliable modules. The results obtained by the GA classifier were also compared with those obtained using Neural Network feed forward classification and GA was found to perform better for the case studied.

As a part of future work, other classification techniques can be applied and the results compared with those obtained in this research. The proposed technique can be applied to selected benchmark applications and the results can improve the credibility of the proposed technique.

REFERENCES


ABOUT THE AUTHORS

S.Suyambu Kesavan M.A, M.Phil., He has 10 years of teaching experience. His area of interest is in the field of Software engineering, Data mining and Soft computing. He is a research scholar at manonmaniam sundaranar university and working as a Programmer Dept. of Computer Science, Sivanthi Aditanar College, Nagercoil.
(email: nellaikesavan@gmail.com)

K.Alagarsamy Ph.D. He is an Associate professor, MCA Dept., School of Information Technology, Madurai Kamaraj University, Madurai. He has 32 years of teaching service. His area of interest in the field of Software Engineering, Data mining and Soft computing. He has published more than 25 papers in various national and international journals.
(email: alagarsamymku@gmail.com)

S.Palanikumar B.E., M.E.,Ph.D., He has 1.5 year of industry experience and 11.5 years of teaching experience. His areas of interest are in the field of Image and Signal Processing, Software Engineering and
Bio-metrics and Embedded and Computer communication Systems. He received his doctoral degree from Anna University, Chennai. And working as an Associate Professor, IT Dept. at Noorul Islam Centre for Higher Education, Noorul Islam University, Kumaracoil. He has published more than 18 papers in national and international journals.
(email:palanikumarcsc@yahoo.com)
A Traceability Approach in Software Creation by DM

S.SuyambuKesavan
Research Scholar
SivanthiAditanar College
PillayarPuram, Nagercoil

Dr.K.Alagarsamy,
Associate Professor
Dept of MCA, ComputerCenter,
Madurai KamarajUniversity, Madurai

Abstract:

In present environment software engineering integrates with data mining. Data mining (DM) works as good decision support system in software tools. Now we discuss about navigation patterns source code tracing design. Static navigation shows constant navigation patterns. Through Static navigation every time execution flow takes same time and identifies same kind of irrelevant data. It cannot provide any reliable solution in some times. It can show the fewer amounts of efficiency and performance.

Present proposed techniques implementation in dynamic navigation. Dynamic navigation tool takes the execution flow amount of time, automatic decision and predictions. In experiment wise shows the best results compare to static navigation.

Keywords: navigation, API, Data mining(DM), Traceability.

Introduction:

Present Users are expects best reliable tools and software systems. Whenever capture some source files to require traceability software. Previously for tracing the files for execution many kinds of issues are generates here. These issues are shows the limitations like reliable, efficiency and performance.

Static navigation shows the execution flow communication with different kinds of source files. Previous source files procedure are handle less amount of unnecessary data only it can handle in implementation. It can take more amounts of execution time and operating system cost. It cannot provide any accurate results. Many number people are expect any final results we are try to implement prototype process.

Now we are proposes one new technique for identification of effective results. That technique is called as dynamic navigation with optimum results.

Below heading are discuss about implementation, design and results

II. Related Work:

In this paper discuss about some modern and traditional applications of software tools. It is not shows any kind of better decision making models implementation. Software tools requires in different number of locations like design, implementation, debugging and testing phases environment process. Present software are not provides any kind of sufficient feedback in different number of locations. Present techniques provide to show the fewer amounts of efficient results in implementation part.

In software’s are implements with reverse and re engineering techniques. Every time consider redesign process. Redesign
identifies as a rework generation. This rework identifies in present tools and generates some kind of inconsistency results specification process. Present and traditional software’s are not maintains the any kind of constraints implementation process. These kind of tools are not provides any reliable solution and accurate solution.

Whenever release the software products in market for identifies the exceptions, bugs and quality with data mining techniques. After defining the some defects related to particular software all kinds of things it can resolve. It is high cost effective products. It is the manual verification techniques.

Previously in many kinds of software’s implemented the software engineering tasks. This process everything focuses on code traces implementation. Exchanges the code amount data through network for verification data mining techniques are implemented. Present verification techniques are not provides any successful solution. Data mining techniques shows the result as a high productivity quality specification.

Operating Systems are also faces the problems in processing stage. Many number of software’s are embedded in operating system. Every times changes the software design and automatically to get the defects in operating system. Frequently changes software design and embedded into operations it is high cost specification results.

In UML tools also contains some problems with only software engineering whenever there is no analysis techniques implementation. It is the prototype way for providing the quality design. Many kinds of engineers and designers are feels like burden. In all situations engineers are think about only design. There is no evidence whenever generate design as a efficient design. No one is not gives the guarantee for providing efficient solution.

In many number of software’s are uses in data warehouse. Data warehouse also contains some kinds of techniques like OLAP techniques implementation process. Present prediction operations are not working properly in implementation process. There is no possibility for modify the query and decision making implementation. All the queries are executes in data warehouse for extract the results using natural language process environment process. There is no possibility for understanding the queries specification process.

The above all the concepts are shows some limitations. These limitations are ready to overcome in new methods and techniques specification.

III. Motivation Analysis:

In software engineering domain using the data mining perform the verification data analysis and discovery data analysis specification process. Data discovery process can starts using navigation patterns. Each and every navigation patterns contains pattern or framework or annotation. Through navigation trace the number of bugs in source code implementation. Half of the bugs are trace through navigation using data mining techniques implementation process. Navigation is possible in different ways of specification process. This paper shows the Comparison in between of static navigation of traceability of bugs to dynamic navigation of traceability of bugs. Dynamic Navigation contains the possibility for trace the high quality bugs trace facility implementation. It can provide good quality, accuracy and performance specification process. Dynamic
Navigation shows the results as a high reliable solution.

**IV. Method implementation:**

Hierarchy based project evolution procedure consider in implementation part. In project evolution stage different phases are available. Each and every evolution phase consider some kind of analysis procedure is require. Each and every analysis procedure using navigation identifies relevant and irrelevant content. Navigation patterns traces and prediction of irrelevant code or unnecessary code. It is possible to remove independent variables and only identifies dependent variables as a final results navigation patterns results. All the dependent variables are works as participators in execution procedure of the project. Execution procedure executes and collects the related data as a clustering data. Clustering data code identifies with sequential association mining methodology and strategy. Clustering code detects with prediction mechanisms. Prediction relates to neural networks and artificial intelligence. Prediction everything it is possible with predefined patterns. All patterns are placed into one API specification process. Each and every pattern API it can update and provide the good guidelines and gets the reliable solution in software development. These are the new features implements with dynamic navigation patterns specification process.

**V. Design and Implementation of Dynamic navigation with data mining under software development tool:**

Dynamic Navigation represents the results with high reusability of code and reliable solution specification. Dynamic navigation identifies the results with classification of classes in software tools. In total software much functionality are present. In execution phase which functionality require that particular part only it can classify that navigation cost is spend very less in implementation. This is possible for detection with pattern mining specification and different domains of results implementation process. It can identify the results as a frequent sequences representation.

**Dynamic navigation of code tracing contains different kinds of phases are present here in implementation process.**

1. Patterns participation
2. Pattern detection with dependent and relationship variables
3. Extraction of pattern Design.
4. Construction of new design specification results as a dynamic navigation tree

**VI. Architecture:**

Fig 1: Complete Flow of Dynamic Navigation patterns Identification
VI.1 Pattern Participation:

Developer forwards one query statement for identification of results. User forwarding patterns starts the detection in pattern database of results using classification and decision trees implementation process for identification of reliable results. Decision tree works with probability for generation of minimized results implementation.

VI.2 Pattern Detection:

Pattern detection classifies the results using different kinds of processing techniques like pre processing and post processing with different kinds of rules implementation. Post processing defines the results as a final amount specification process.

VI.3 Extraction of pattern Design:

Using dependent and independent variables specification identifies the relationship and checks the validation is this pattern comes under perfect pattern or not using API pattern or framework pattern. These kind of patterns are comes to sequential pattern or association pattern based results identification process.

VI.4 Construct of new design as a dynamic Navigation tree:

According to user expected features changes the structure of annotation based procedure as a final output identification process. It can provide as a quality pattern specification process. These kind quality patterns traces in software tool as a dynamic reusability specification process. It can spend the less amount of operating system cost and I/O cost specification also here in implementation.

VII. Experimental tool with my eclipse 10.5:

Experiment starts in my eclipse 10.5. All architecture based steps are implemented for tracing the source code patterns. Tool identifies different kinds of tasks in implementation procedure. Tasks are shown in execution flow specification process. Execution flow with classification identifies the related source files of content. All the sources files are displayed as final results in implementation process. Final results are collects using association rules implementation and generate the sequential patterns of content. It can takes less navigation time and cost in software development and redesign specification process. According user flexibility tree of concepts are try to change in implementation process. Navigation cost reduces with good data mining techniques implementation only.

VIII. Results and Discussions:

It can show the one guide development tool for removing the unnecessary files of content. It can trace the source files in existing environment as static files based navigation environment process. We are implemented successfully as dynamic navigation based files...
and provides the successful solution. It can takes amount of execution time and provides good reliable solution in implementation of new tool.

IX. Conclusion:

Mining and software engineering provides good tracing of files in implementation. It explains difference in between of static trace files and dynamic trace implementation process. It can reduce the navigation cost and execution of flow of time also in implementation. It can collect the exact source files and displayed as a final output sequential pattern specification results. User expected all the source files also located into pattern itself.

X. References:

1. A SYSTEMATIC STUDY OF SOFTWARE QUALITY MODELS
2. Introducing Data Mining Techniques and Software Engineering to High School Science Students
3. Do Software Engineers Benefit from Source Code Navigation with Traceability?
4. Predicting Software Escalations with Maximum ROI
5. Mining Software Engineering Data
6. Data Mining for Secure Software Engineering – Source Code Management Tool Case Study
7. Data Mining for Software Process Discovery in Open Source Software Development Communities
8. Software Specification Discovery : A New Data Mining Approach
9. Software Escalation Prediction with Data Mining
10. Mining Source Codes to Guide Software Development
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 no stack trace of the failure is available. With occasional bugs, the situation is evenmore difficult, as they are harder to reproduce. Developers usually try to find and rectify bugs by doing an in depth code review along with testing methods and classical debugging strategies. Since such reviews are very expensive, there is a need for tools which localise 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.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEOLCommentToken</td>
<td>End of line comment.</td>
</tr>
<tr>
<td>CNumericToken</td>
<td>Any Numeric value</td>
</tr>
<tr>
<td>CEOF</td>
<td>Token End of file.</td>
</tr>
<tr>
<td>CEOL</td>
<td>Token End of line.</td>
</tr>
</tbody>
</table>

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:**


Data Mining Approach in Software Analysis

S.Suyambu Kesavan
Research Scholar
Sivanthi Aditanar College
Pillayar Puram, Nagercoil

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

Abstract:

Data mining and knowledge discovery have proved to be valuable tools in various domains such as production, health care and management. Data mining also has potential to address some highly challenging areas of software engineering such as adaptability and security. In software engineering process analyst play an important role for gathering information from the statement of user and obtaining the information from many resource. Data mining gives the potential algorithms and resource for collecting the information. In this paper we are merging the concept of data mining algorithms into software engineering techniques to collect the information and produce the better analyst decision.

Keywords: Software engineering, Data Mining, Clustering algorithm.

Introduction:

Data mining:

Data Mining, also popularly termed as Knowledge Discovery in Databases, refers to the nontrivial extraction of implicit, previously unprocessed and potentially useful information from data in databases. While data mining and knowledge discovery in databases are frequently treated as synonyms, data mining is important part of the knowledge discovery process. The Knowledge prediction in Databases process comprises of a few steps leading from raw data collections to some form of new knowledge. The iterative process consists of the following steps:

- **Data cleaning:**
  It is a phase in which noise data and irrelevant data are removed from the collection.

- **Data integration:**

At this stage, multiple data sources, often heterogeneous, may be combined in a common source.

- **Data selection:**
  The data relevant to the analysis is decided on and retrieved from the data collection.

- **Data transformation:**
  It is a phase in which the selected data is transformed into forms appropriate for the mining procedure.

- **Data mining:**
  It is the critical step in which clever techniques are applied to extract patterns potentially useful.

- **Pattern evaluation:**
  In this step, strictly interesting patterns representing knowledge are identified based on given algorithmic procedure.

- **Knowledge representation:**
  This is the final phase in which the discovered knowledge is visually represented to the user. This step uses visualization techniques to help users understand and interpret the data mining results.

It is usual to combine some of these steps together. For instance, data cleaning and data integration can be merged together as a preprocessing phase to generate a data warehouse. Data selection and data transformation can also be merged where the consolidation of the data is the result of the selection or as for the case of data warehouses; the selection is done on modified data.

Software Engineering:

Software Engineering is a technique dedicated to designing, implementing, and modifying software so that it is become of high quality, affordable, maintainable and fast to
build. It is a step by step approach to the analysis, design, assessment, implementation, test, maintenance and reengineering of software development that is the application of engineering to software.

There are three main activities to be performed before to the start of software like planning, creating the stakeholder requirements, and defining and deploying the development environment. Once these activities are completed, we are ready to initiate the project. The project in run as a series of incremental development efforts, each expanding and elaborating on the efforts that came before. Some of the important points are:

- **Project Initiation**
  Setting up the team along with development environment, Understanding what the customer needs, planning the project

- **Perspire planning**
  Parallel activities may be done by different people; Parallel activities may be done in any order with respect to each other.

- **Creating the Schedule**
  Steps involved in this task are identifying the desired functionality, identifying the key risks

- **Creating the team work**
  There exists a strong correlation between the team structure and the model organization. Teams are formed because they make coherent sense and model is organized to allow the teams to work together effectively.

- **Planning for reuse**
  Identifying reuse needs and goals, Identifying opportunities for reuse, Estimates the cost of constructing reusable assets, determining which reusable assets to construct

- **Specifying logical architecture**
  This is also known as project structure or model organization involving specification of model organization patterns and checklist for logical architecture.

- **Performing the initial safety and reliability analysis**
  Steps involved are identify the hazards, quantify the problems in terms of likelihood and severity, Compute the risks and perform an initial safety analysis.

  These are some of the key points where software analyst play a vital role for collecting the information. Information gathering is a huge process and tedious process. Information source may any form like document, ppt, pdf, etc. If it is manual search it may lead to some problem and accuracy problem due to that we are moving some other automation technique.

**Data mining for SE:**

Mining software engineering data has emerging as a research direction over the past years. This research direction has already achieved substantial success in both research and practice. In this paper, we declare Software Intelligence as the future of mining software engineering data, within modern software engineering research, practice and education. The vision of Software Intelligence (SI) has yet to become a reality. Nevertheless, recent advances in the Mining Software Repositories field show great promise and provide strong support for realizing SI in the near future, as software engineering research aims to ensure its relevance and impact on modern software practice. This position paper summarizes state of practice and research of SI, and lays out future research directions of mining software engineering data tenable SI.

Text mining is a new and exciting research area that attempts to solve the information overload problem. It uses many techniques from data mining, but since it deals with unstructured data, a major part of the text mining process deals with the crucial stage of...
preprocessing the document collections. The process also involves the storage of the intermediate representations, techniques to analyze these intermediate representations. A typical text mining system begins with collections of raw documents, without any labels or tags. Documents are then automatically tagged by categories, terms or relationships extracted directly from the documents. Next, extracted categories, Entities and relationships are used to support a range of data mining operations on the documents. In this paper we are going to implement the innovative approach of data mining in software engineering for helping he software analyst.

Proposed method:

In our research we are going to incorporate the probabilistic clustering method for our analyst usage. This is the combination approach of software engineering analysis phase with data mining clustering approach. Here we will discuss about the characteristics probability clustering approach, data is considered to be a sample independently taken from the user objectives of several probability distributions. The main idea is that data points are generated by first randomly taken a point x from a corresponding distribution. The area around the mean of each distribution constitutes a natural cluster. So we associate the cluster with the corresponding distributions parameters such as mean, variance etc. Each data point carries not only its attributes but also a cluster ID. Each point x is assumed to belong to one and only one cluster and we can estimate the probabilities of the assignment.

Customer objective serves as an objective function, which gives rise to the Expectation Maximization (E-M) method is a two-step iterative optimization. Step (E) estimates probabilities, which is equivalent to a soft reassignment. Step (M) finds an approximation to the customer objective given current soft assignments. This boils down to finding mixture model parameters that maximize estimation result. The process continues until estimation convergence is achieved.

We can also use some other tricks to facilitate finding better local optimum suggested acceleration of EM method based on a special data index, decision tree, KD-tree, etc. Data is split at each node into two descendants by dividing the widest attribute at the center of its range. Every node stores sufficient statistics. Approximate computing over a pruned tree accelerates EM iterations.

Probabilistic clustering has some important features:

- It can be altered to handle recodes of complex objective of the user.
- It can be stopped and resumed with consecutive batches of data, since clusters have representation totally different from sets of points
- At any stage of iterative process the intermediate mixture model can be used to assign cases (on-line property)
- It results in easily interpretable cluster system

Because the mixture model has clear probabilistic foundation, the determination of the most suitable number of clusters k becomes a more tractable task. From a data mining perspective, excessive parameter set causes over fitting, while from a probabilistic perspective, number of parameters can be addressed within the Bayesian framework.

Result Discussion:

We have taken set of sample user objectives of the software engineering for analysis purpose. We tested performance effectiveness with the human analyst with our probabilistic clustering method. Below diagram shows the effectiveness of the probabilistic clustering method. The result value is given out of ten.

<table>
<thead>
<tr>
<th></th>
<th>Analyst</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missed Requirements</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Accuracy</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Remembrance</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
Conclusion:

In past decade, lot of innovative approach method has been implemented in software engineering and data mining, which yields the highly valuable solutions for software industry. In our research we carried out the data mining with software engineering approach for analyst purpose. Compare with human analysis it will give more accuracy and also it reduces the human effort and counting. It’s our belief, our work will highly helpful to the software engineering society.

Future Work:

In our research we carried out probabilistic clustering method to form a clustering analysis of user objective. There are many clustering algorithm present in data mining domain. In future we can use some other algorithm for the implementation and deriving the conclusion from comparative analysis. We can also predict which one is the best algorithm for software engineering.

Reference:


