ISSN:0975-9646

Improving Software Development Process through Data Mining Techniques Embedding Alitheia Core Tool

Ms Anupama Das[1], Ms.Kaberi Das[2], Prof (Dr) B.Puthal[1]

[1]Department of Computer Applications,[2] Department of Computer Science and Engineering, Shiksha 'O' Anusandhan University Jagmohan Nagar, Bhubaneswar, 751030, INDIA

***Abstract*—Research in the fields of software quality, maintainability requires the analysis of large quantity of data, which originate from software projects. It is a challenging task in pre-processing the data and synthesizing the composite results. It is very often an error prone task. Data mining techniques are generally considered as the best for pre-processing data. But there may be case that data is not up to the mark for that, in this context an improvised core tool will be used to facilitate software engineering research on large and diverse data set .There may be the chance of not getting the proper result for boosting up the software industry, but upto certain extent it will be surely helpful in research field or to the researchers for further innovative development.**

***Keywords*— Data mining, Software engineering, Feature selection, Alithea core tool.**

I. INTRODUCTION

Software development in earlier times was focused on creating algorithms and implementing through programs. This technique was capable of solving almost all software programs but evolution of sophisticated hardware required continuous project planning resulted in low productivity, heavy maintenance cost which led to failure of user expectation and led to stagnation of software development. Slowly it caused software crisis and highlighted in NATO conference in 1968.This crisis was caused by the fact that were no formal method and methodologies support tools for project management.

Software community realized that to solve large software intensive system, they borrowed ideas from other fields of engineering and incorporated into software development. This was the origin of software engineering. Software engineering is a profession dedicated to designing, implementing, and modifying software so that it is of higher quality, more affordable, maintainable, and faster to build. The IEEE Computer Society's Software Engineering Body of Knowledge defines "software engineering" as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. The study of these approaches; led to the application of engineering tools in software development.
Knowledge discovery process in databases (KDD) borrowed from data mining gave rise to data mining algorithm that were capable of solving many complicated software problems. Data mining, a branch of computer science is the process of extracting patterns from large data sets by combining methods from statistics and artificial intelligence with database management. Data mining is seen as an increasingly important tool by modern business to transform data into business intelligence giving an informational advantage.

Software engineering data (such as code bases, exe-cution traces, historical code changes, mailing lists, and bug databases) contains a wealth of information about a project's status, progress, and evolution. Here objective is to refine these data trough an algorithm so that it can be processed to get a well defined software process. Algorithms are applied so that the complexity, time delay and cost can be reduced. This can be done by putting the data into the format for further processing in subsequent stages.

Using data mining technique we will explore the potential of this valuable data in order to better manage in subsequent stages of development and produce high quality software systems, which can be delivered on time and within stipulated budget.
Mining algorithms fall into four main categories:

• Frequent pattern mining—finding commonly occurring patterns;
• Pattern matching—finding data instances for given patterns;
• Clustering—grouping data into clusters; and
• Classification—predicting labels of data based on already-labelled data.[1]

II EXISTING LAYER MODEL

Firstly existing model gives rise to the challenges of mining software engineering data. Second, development frontier of data mining practice in software engineering.Third, intended to analyze successful cases of mining SE data. Finally, intended to give an overview on commonly used data mining tools. Our overview will help the participants gain a better understanding of available tools. The participants can use such tools in order to explore their data and integrate data mining techniques in their research and day to day work.

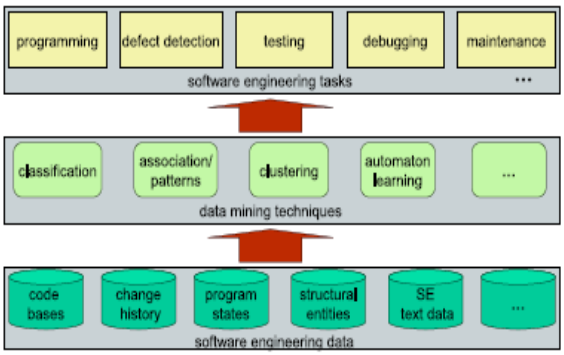

Fig 1: OVERVIEW OF MINING SE DATA

Existing model in Fig 1 has got certain drawbacks for which developers are facing problems in conducting research with software repository.

The main problem developers face working with software repository is that it is difficult to set up experiments that can be extended or replicated as a result of which they invariably face hurdles which the researcher must overcome in order to experiments with huge data chunks from large number of projects. To circumvent these hurdles we want to use a Alitheia core tool which is an extensible platform for software quality analysis. It is designed specifically to facilitate software engineering research on large and diverse data sets .By Alitheia core tool, integrating data collection and pre-processing phases with an array of analysis services and subsequently presenting the researchers with an easy to use extension mechanism. So Alitheia core tool can be the basis of shared tool and research data that will enable researchers to focus on their research questions at hand rather than spend time on reimplementing analysis tool.[10]

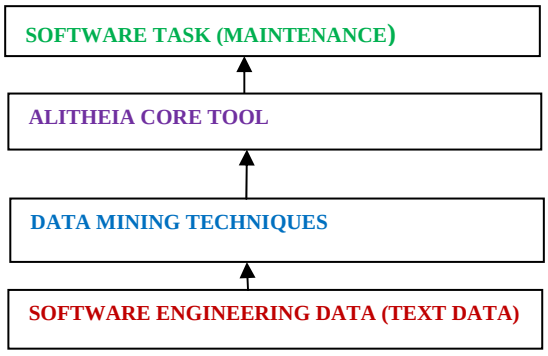

Fig 2: PROPOSED LAYER MODEL.

A. Software Engineering Data

SE text data include bug reports, e-mails, code comments, and documentation for API methods. Common types of text mining algorithms include text clustering, classification, and matching. Example text clustering applications include clustering bug reports to detect duplicate bug reports and thereby reduce inspection efforts, and assigning reports to specific developers to fix the bugs. Example text classification applications include recommending assignment of a new bug report to a specific developer based on the past assignment of old bug reports. Example text matching applications include searching keywords in code comments, API documentation, or bug reports, and detecting duplicates of a given bug report among old reports.[2]

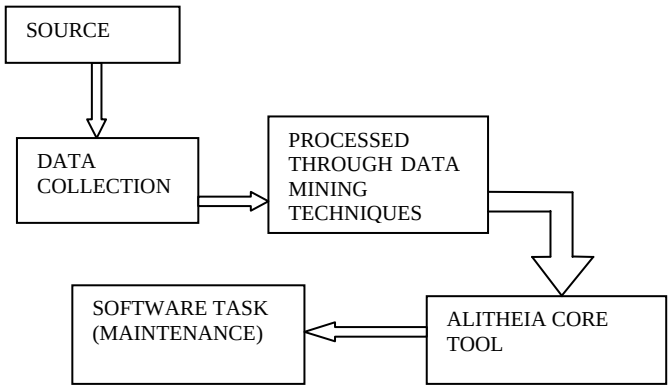

Fig 3: WORKFLOW OF PROPOSED MODEL

Software engineering data may have missing data. Missing data analysis is a wide research area over the past 30-35 years. There are three approaches to this problem. First, there are missing data ignoring techniques, e.g. (Haitovsky, 1968; Roth, 1994). Second, there are missing data toleration techniques (Aggarwal and Parthasarathy,2001;Schuurmans and Greiner, 1997). Third, there are missing data imputation techniques which are the emphasis of this paper, e.g. (Friedman, 1998; Little, 1988; Schafer and Olsen, 1998; Shirani et al., 2000; Troyanskaya et al., 2001)

The missing data ignoring techniques simply delete the cases that contain missing data Because of their simplicity, they are widely used (Roth, 1994) This approach has two forms:

1) *List wise deletion (LD):* is also referred to as case deletion, case wise deletion or complete case analysis. This method omits the cases containing missing values. It is easy, fast, does not 'invent data', commonly accepted and is the default of most statistical packages. The drawback is that its application may lead to a large loss of observations, which may result in too small data sets if the fraction of missing values is high and particularly if the original data set is itself small, as is often the situation for software project estimation (Myrtveit et al., 2001).

2) *Pair wise deletion (PD)*: is also referred to as the available case method. This method considers each feature separately. For each feature, all recorded values in each observation are considered (Strike et al., 2001) and missing data are ignored. This means that different calculations will utilise different cases and will have different sample sizes, an undesirable effect. The advantage is that the sample size

for each individual analysis is generally higher than with complete case analysis. It is necessary when the overall sample size is small or the number of cases with missing data is large.

The missing data toleration techniques use a probabilistic approach to handle missing data. They do not predict missing data but assign a probability to each of the possible values. Thus they are internal missing data treatment strategies, which perform analysis directly using the data set with missing values. The missing data imputation techniques estimate missing values for the missing cases and insert estimates obtained from other reported values to produce an estimated complete case. The common forms are as follows:

1) *Mean imputation (MI):* is also referred to as unconditional mean imputation. This method imputes each missing value with the mean of reported values. The disadvantage is that it leads to underestimation of the population variance

2) Regression imputation (RI): is also referred to as conditional mean imputation .The regression model is built using the complete observations. It tends to perform better than MI, but still underestimates variance.

3*) Hot-deck imputation (HDI)*: methods fill in missing data by taking values from other observations in the same data set.. The choice of which value to take depends on the observation containing the missing value. Randomly choosing observed values from donor cases is the simplest hot-deck method.

4) *Multiple imputations*: means that the missing data are imputed m > 1 times, with a different randomly chosen error term added in each imputation. In this method, each missing value is replaced by a set of m plausible values drawn from their predictive distribution. After performing multiple imputations, there are m complete, imputed data sets [3].

B) Feature Selection Method

Feature selection is a pre-processing technique commonly used on high dimensional data. Feature selection studies how to select a subset or list of attributes or variables that are used to construct models describing data. Its purposes include reducing dimensionality, removing irrelevant and redundant features, reducing the amount of data needed for learning, improving algorithms' predictive accuracy, and increasing the constructed models' comprehensibility.

Feature selection extracts the most important set of attributes for model training. a feature-selection algorithm is part of the classification rule. This is why feature selection must be included when using cross-validation error estimation

Feature Selection for Unlabeled Data: Unsupervised learning, or cluster analysis, aims to group similar objects Many clustering algorithms assume that domain experts have determined relevant features. But not all features are important; some But not all features are important; some might be redundant or irrelevant. And the presence of many irrelevant features can even misguide clustering results. Moreover, reducing the number of features increases comprehensibility and ameliorates the problem of some unsupervised-learning algorithms failing with high-dimensional data [9]We examine the feature selection in the context of software quality estimation (also referred to as software defect prediction), where a classification model is used to predict program modules (instances) as fault-prone (fp) or not-fault-prone (nfp)([4],[5],[6])

Such a model is usually trained using software measurement and defect data from previous development experiences, and the model is then applied to predict the quality of target (under-development) program modules. As a result, practitioners can strategically allocate project resources and focus more on those potential problematic modules. The process of using feature selection provides a different training data set for building the classification models. Typically, feature selection techniques are divided into two categories: wrapper-based approach and filter-based approach. The wrapper-based approach involves training a learner during the feature selection process, while the filter-based approach uses the intrinsic characteristics of the data, based on a given metric, for feature selection and does not depend on training a learner. The primary advantage of the filter-based approach over the wrapper-based approach is that it is computationally faster. Saeys et al. [7] investigated the use of ensemble feature selection techniques, where multiple feature selection methods were combined to yield results. Liu et al. [8] introduced the concept of active feature selection, and investigated a selective sampling approach to active feature selection in a filter model setting.

C) Alitheia Core Tool

Now coming towards the software engineering technique we will using Alitheia core tool for software quality analysis that is designed specifically to facilitate software engineering research on large and diverse data sources, by integrating data collection and pre-processing phases with an array of analysis services, and presenting the researcher with an easy to use extension mechanism. Alitheia Core aims to be the basis of an ecosystem of shared tools and research data that will enable researchers to focus on their research questions at hand, rather than spend time on reimplementing analysis tools. The Alitheia Core has been designed from the ground up for performance and scalability Alitheia Core also includes clustering capabilities through the cluster service. The development of the cluster service was based on the observation that after the initial metadata synchronization, the workloads on the system processes are usually embarrassingly parallel. The cluster service restricts access to projects during metadata updates and allows metrics to be run on several nodes in parallel..

The Alitheia Core itself is not concerned with mirroring data from projects; it expects data to be mirrored externally.

This choice was made at the beginning of the project to compensate for the large number of different data sources which the system should work with The Alitheia Cores system is a platform modelled around a pluggable, extensible architecture that allows it to incorporate many types of data sources and be accessible through various user interfaces.[10]

D) Software Maintenance

Software maintenance is one of the major concerns of software development and maintenance organizations. "Software maintenance is the process of modifying a software system or component after delivery to correct faults, improve performances or other attributes, or adapt to a changed environment."[12] Although the software maintenance phase starts after the delivery of product to the client, it covers a major portion of the cost, effort and time involved in the project

1) Issues in Software Maintenance:

Impact Analysis, Complex code and Architecture, Lack of Understanding, Undefined Process and Procedure for Maintenance, Involvement of Senior Staff, Maintenance Cost Estimation, Improper User Training, and Dependency on Outside Supplier, Lack of Documentation, Measuring Maintenance and Support Service. Motivation of Support personal.[13]

2) Managing Software Maintenance Costs by Developmental Techniques and Management Decision during Envelopment

1. Strive for Commonality2. Apply Industrial Engineering Practices to Software3. Engage 4. Adopt a Holistic Approach to Sustainment 5. Develop Highly Maintainable Systems and Software6. Manage the Off-the-Shelf Software7. Plan for the Unexpected8. Analyze and Refine the Software Sustainment Business Case (use Parametric software sustainment cost estimates)[14]

Software maintenance deals with the management of such changes, ensuring that the software remains correct while features are added or removed. Maintenance cost can contribute up to 60–80% of software cost A challenge to software maintenance is to keep documented specifications accurate and updated as the program changes.

Outdated specifications cause difficulties in program comprehension, which account for up to 50% of program maintenance cost. [11]

IV. FUTURE WORK

More emphasis can be given in data collection, data pre-processing, and feature selection. There can be some advanced software engineering technique for pre-processing data. We can also combine data mining technique and software engineering technique for better pre-processing. We should also concentrate on software maintenance cost. Alitheia core system can be modelled as an extensible architecture allowing incorporating many types of data sources and accessible through various user interfaces .Future plan work on this platform includes development of plug –ins for support of other sources like web services and full fledged API for accessing project meta data in simple way.

V. CONCLUSION

Intensive software project requires handling huge number of data. Data can be text data .it is necessary to pre-process the data through data mining technique. For better pre-processing we can utilize Alitheia Core Tool .It may or may not be useful for software industry but it will be surely beneficial for research field.

REFERENCES

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000

[2] Tao Xie and Suresh Thummalapenta, North Carolina State University David Lo, Singapore Management University Chao Liu, Microsoft Research "*Data mining for software engineering*"

[3] *A new imputation method for small software project data sets* Qinbao Song a,*, Martin Shepperd b a Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China b Brunel University, Uxbridge, UB8 3PH, UK Received 17 March 2005; received in revised form 30 April 2006; accepted 3 May 2006 Available online 16 June 2006

[4] Y. Jiang, J. Lin, B. Cukie, and T. Menzies. *Variance analysis in software fault prediction models.* In Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering, pages 99–108, Bangalore-Mysore, India, Nov. 16-19 2009.

[5] T. M. Khoshgoftaar, P. Rebours, and N. Seliya. *Software quality analysis by combining multiple projects and learners.* Software Quality Journal, 17(1):25–49, March 2009.

[6] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. *Benchmarking classification models for software defect*, prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 34(4):485– 496, July-August 2008

[7] Y. Saeys, T. Abeel, and Y. Peer. *Robust feature selection using ensemble feature selection techniques.* In Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II (2008), pages 313–325, 2008.

[8] H. Liu, H. Motoda, and L. Yu. *A selective sampling approach to active feature selection. Artificial Intelligence*,159(1-2):49–74, November 2004

[9] *"evolving feature selection*". Huan Liu, Arizona State University.

[10] *A Platform for Software Engineering* Research Georgios Gousios, Diomidis Spinellis Department of Management Science and Technlology Athens University of Economics and Business Athens, Greece gousiosg,dds@aueb.gr

[11] Canfora G, Cimitile *A. Software maintenance. Handbook of Software Engineering and Knowledge Engineering. World Scientific:* River Edge NJ, 2001; **1**:91–120

[12] IEEE Std. 610.12, *"Standard Glossary of Software Engineering Terminology"*, IEEE Computer Society Press, Los Alamitos, CA,1990

[13] *A Framework for Software Maintenance and Support Phase* Zafar Nasir and Abu Zafar Abbasi Department of Computer Science National University of Computer & Emerging Sciences Karachi, Pakistan{zafar.nasir, abuzafar.abbasi}@nu.edu.pk

[14] *Software Maintenance Implications on Cost* and Schedule Bob Hunt, Bryn Turner, Karen McRitchie Galorath Incorporated 100 North Sepulveda Boulevard Suite 1801 El Segundo, California 90245(703) 201-0651 bhunt@galorath.com

[15] *Mining Software Engineering Data* Tao Xie North Carolina State Univ. USA xie@csc.ncsu.edu Jian PeinSimon Fraser Univ. Canada jpei@cs.sfu.ca Ahmed E. Hassan Univ. of Victoria Canada ahmed@ece.uvic.ca