Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

International Conference on Mobility in Computing- ICMiC13, Organized by Mar Baselios College of Engineering and Technology during December 17-18, 2013 at Trivandrum, Kerala, India, pg.1 – 7

RESEARCH ARTICLE

Data Mining for Causal Analysis of Software Defects

Remya Kartha K[1], Vikraman Nair R[2]

Department of Computer Science and Engineering, Mar Baselios College of Engineering and Technology, Trivandrum, India

[1] remya.kartha@gmail.com; [2] rvnair1951@gmail.com

***Abstract*—Despite careful planning, clear documentation and proper software process control, software defects are inevitable. Defect causal analysis is the nucleus of process improvement approaches. Causal analysis is regarded as one of the core steps in software defect prediction. The analysis of software faults improves the reliability and quality of software products thereby adding value to the business. Many data mining techniques have been suggested for defect prediction and prevention. This paper summarizes the various data mining techniques used in defect prediction and defect prevention and also proposes the development of a data mining based tool to automate the root cause analysis of software defects. The proposed tool makes use of Hadoop File System to store the defect data. Implementation of decision tree algorithm in MapReduce is proposed for the classification and analysis of defects.**

Keywords— Software Fault Analysis; Root Cause analysis; Defect Classification; Defect Categorization

I. INTRODUCTION

Software development is a human activity and it is impossible to deliver completely defect free products. An error, fault or failure in a computer system or computer program that generates an inaccurate or unanticipated result is termed as a defect or bug. Bugs can happen due to mistakes and errors made by humans in requirement elicitation, design, coding, frameworks and operating systems used by programs or by compilers producing incorrect code. Software defects can trigger errors that can in turn have a variety of ripple effects, with varying degrees of severity. Some defects have a subtle effect on the functionality and can remain undetected for a very long time. More serious defects can cause the program to crash, lead to catastrophic failures or even result in loss of life. One of the most expensive activities of the software development life cycle is the cost of finding and rectifying the defects. Software engineering organizations are reluctant to admit failures and usually do not put in much effort to investigate them. Analyses of failures lead to improvements in engineering practice [7]. Defect data can give valuable insight into the efficiency of the software development process. Development teams rarely analyse this asset to its full potential. The total quality cost can be decreased manifold by even a small increase in the defect prevention practices.

Causal Analysis is a major component of modern process improvement approaches, e.g., the CMMI, Six Sigma, and Lean [6]. Software Root Cause Analysis is the method of systematically organizing and analysing the defects recorded during software development. Through defect causal analysis it is possible to identify the fundamental cause of project problems [5] and use this information as a feedback mechanism to improve the productivity and quality of software development process. Understanding the type of defects and managing repetitive defects can help in suggesting improvement actions like deployment of automatic

bug finding tools, rigorous testing, developer training etc. Diagnosing the cause of faults in large software systems is a challenging task. Marcos Kalinowski *et al* [5] have presented a set of guidelines for causal analysis of software defects which emphasize the importance of defect cause analysis in software industry. Defect Causal Analysis (DCA) provides a means for improvement of software process based on defect data.

Data Mining is the process of collecting data from different perspectives, analysing the data and summarizing it into useful information. The process of data mining enables the users to understand the relationship between data. Data mining reveals trends and patterns hidden in the data. Data mining techniques can be applied to historical defect data to identify the most frequently occurring causes of defects. This information can be applied to software quality improvement process. Data is extracted from software repositories. Data mining techniques are applied to the extracted data to identify patterns of defects and their causes. The data mining techniques commonly used in causal analysis and defect prediction are classification, clustering and association mining. Classification is a data mining technique that assigns items in a collection to target classes. Once classes have been identified, the system infers rules that govern these classes. This is a form of supervised learning. Classification approaches make use of a training set where all data is associated to a known class. The classification algorithm learns the training set and creates a model based on its learning. The model is used to classify unknown data. In software engineering, classification approaches have been used in building software prediction models. The training set consists of defects whose classes are already known. This information is used to train the classifier to identify similar defects. The model can then be used to predict defects. Clustering is an unsupervised data mining technique in which class labels are not already known. In this approach data items are grouped on the basis of how similar they are to other data items. Clustering is a process of grouping data such that data items which are similar are put into the same cluster. K-mean clustering is one clustering algorithm frequently used in software defect prediction. In this technique, defect data items are moved among clusters until the most suitable cluster is found. This approach is applied for predicting faults in program modules. Association Mining is a data mining approach for identifying frequently occurring data item sets. This is a method of finding correlations between items in a data set. The Association mining technique first identifies sets of frequently occurring data items. The second step in association mining involves testing the item sets and generating rules for association. The rules are based on a high confidence level. These rules imply how a subset of items influences the occurrence of another subset. In software defect prediction, defect data type is used to predict defect associations among different defect types. The defect associations help us to predict defects related to the detected defects and take suitable corrective measures.

The rest of the paper is organized as follows: Section II presents a review of the data mining approaches used in software defect prediction and prevention. Section III elaborates the stages in root cause investigation process. Section IV describes the implementation approach for the proposed root cause analysis tool. Section IV and V provide the conclusion and references respectively.

II. **RELATED WORK**

This section reviews some of the data mining approaches employed in software defect prediction and prevention models.

A decision tree based defect classification approach has been adopted in [2]. After the identification of defects, the proposed system classifies defects based on the ID3 (Iterative Dichotomiser 3) decision tree algorithm. The dataset is given as input to the system. The volume, program length, difficulty, effort and time estimator attributes are used to classify the defects as Blocker Type, Critical Type, Major Type, Minor Type and Trivial Type. Decision tree is generated from the training sample set. New instances of defects are classified starting at the root node of the tree, testing the attribute specified at the node and moving down the tree branch depending on the value of the tested attribute. When a leaf node is reached, the defect has been classified. Defect patterns are then identified by employing pattern mining approaches. The pattern mining approach used is that of association rule mining. The final step of the proposed method is to make use of quality metrics like defect density and Defect Removal Efficiency to give an assurance on the quality.

IBM's Orthogonal Defect Classification (ODC) is used to classify and analyse defect data to improve software development [3][4]. Orthogonal Defect Classification makes use of defect data as the source of information. This technique involves the use of statistical methods to predict the phase of the project in which the defect originated. ODC is based on the principle that different types of defects originate from different phases in the project. If there are large numbers of defects in a particular phase of the project, it indicates an area which requires more attention. ODC makes use of three attributes: Defect Type, Defect Trigger and Defect Impact. Defect Type is used to characterize the defect based on the type of change that is required to fix the defect. It is a measure of the progress of the product through the development process. Defect Trigger characterizes the defect based on the factor that created the defect failure. It is a measurement of the verification process. Defect impact measures the impact of the defect on the customer.

Methodologies like Orthogonal Defect Classification, Iteration defect reduction, capturing defects at early stages have been combined in the defect prevention approach proposed in [8]. Information regarding number of lines of code (KLOC) and number of defects is used in the calculation of defect density. The defect density calculated is used to compare the impact of

defect reduction on projects that have implemented defect prevention measures; with the impact on projects that have not implemented any defect prevention. Pareto charts are drawn to identify defect types with highest frequency of defects. The root causes of defects are then identified using a fish bone diagram. The causes identified are discussed with the team and the main reasons for causes are finalized. The set of preventive actions for these causes are determined.

Another data mining approach used in defect prediction models is the subgroup discovery. Subgroup Discovery deals with the discovery of statistically distinct subgroups based on some property of interest. Rules are used to represent subgroups. SD algorithms can be both predictive and descriptive. In the predictive approach rules are determined on the basis of historical data and a property of interest. The descriptive approach of SD algorithms are used for discovering interesting patterns in the data .A defect prediction model that characterizes defective modules on the basis of SD algorithms has been proposed in [12]. They have described the use of the subgroup discovery algorithms: SD algorithm and CN2-SD to identify fault prone modules. Publically available datasets were analysed independently using the SD and CN2-SD algorithms. The study showed that the rules are capable of correctly characterizing the defects based on defective modules. The experiment was carried out in three stages. In the first stage, the data sets were analysed separately. The induced rules increase the probability of finding defective modules. In the second stage, the datasets were combined and the algorithms were verified to ensure that correct results were obtained. In the third stage, the rules were analysed to see if they could be generalized to be used with new data that was part of the training set.

A group of classifiers can significantly improve the classification performance compared to a single classifier. The defect prediction model proposed in [13] makes use of classifier ensemble. The classifications methods used are Bagging, Boosting, Random trees, Random forest, Random subspace, Stacking and Voting. The performance of classifier ensemble approach is compared with that of a model using Naïve Bayes.

III. **WORK FLOW STAGES IN ROOT CAUSE INVESTIGATION PROCESS**

Root Cause Analysis is crucial to software defect prediction and software defect prevention. Root cause identification helps prevent defect recurrences in future and also aid in prediction of defects. This paper proposes a tool for automating the root cause analysis of software defects. The work flow stages involved in Root Cause Investigation process of the proposed tool are Defect Identification, Defect Classification and Root Cause Analysis of Defects.

A. Defect Identification

Defects can be identified at various stages of software development through approaches like design review, code inspection, prototypes, testing and correctness proof. Inspection is the process of examining human artefacts to discover defects in early stages of development. It is regarded as the most efficient and effective quality assurance technique. A prototype is an experimental model of a software product or information system. It enables both the developers and the customers to verify that the design meets all the requirements. A review is a formal session or meeting during which either the design or the product is examined by project team members, managers, users or customers for comment or approval. Testing is an activity aimed at evaluating the capability of a program or software system. The main aim of testing is to uncover errors which could have escaped during investigations in early stages of development. Correctness proofs detect errors introduced during coding. Code which does not meet the requirements of the proof indicates defects. Another valuable source of defect data is the application logs. Some organizations rely on Word documents and Excel files to document identified defects. Few widely used defect tracking tools include JIRA, Bugzilla, Trac etc.

B. Defect Classification

Once the defects have been identified and documented, the next step is the classification of defects. There are a number of ways defects can be classified. Defects can be classified into a variety of types or classes based on the nature of defects. IEEE Std. 1044-2009: IEEE Standard Classification for Software Anomalies classifies defects primarily into the following types [10]:

- Logic problem
- Computation problem
- Documentation problem
- Document quality problem
- Interface/timing problem
- Enhancement
- Data handling problem

- Failure caused by a previous fix
- Data problem and other problem

Defects can be classified based on the phase of detection. There are four stages of a software life cycle where a defect is likely to be detected. These stages are [9]

- During development
- During Testing
- During Normal Operation
- Enhancement or Corrective Maintenance

Another defect classification method is based on the life cycle stage or activity which introduced the defect.

Two of the very commonly used defect classification approaches are IBM's Orthogonal Defect Classification (ODC) and HP's Defect Origin's, Types and Modes [11]. ODC aids in categorizing the defects into classes that cooperatively point to parts of the process which need attention. The ODC approach classifies defects based on three dimensions. The dimensions are defect origin, defect type and defect mode. The HP model links together the defect type to origin and mode. For example a user interface type defect originated in the design phase due to missing specifications. Here the type of the defect is user interface type, origin is design phase and mode is due to missing specifications. Some organizations classify defects based on factors like logical functions, user interface, maintainability, standards and so on. Each company has its own scheme of defect classification.

C. Defect Analysis

There are typically two techniques used in the analysis of data collected from software defects: Statistical Defect Modelling and Root Causal Analysis (RCA). In Statistical Defect Modelling, each defect is considered as a sample from a collection of identified defects. A statistical model is fitted to the collection of identified defects. Counting techniques, comparisons with chronological data and reliability modelling are some of the methods used in the statistical modelling approach. In Casual Analysis, defect is considered to be a unique occurrence and attempt is made to find out the primary cause of each defect.

There are a number of Root Cause Analysis Tools and Methods in use today. A tool has limited use while a method usually involves a number of steps and processes and is used extensively. Some of the graphical tools used in Root Cause Analysis are the Pareto Diagram, Fishbone Diagram, and Fault Tree. The Root Cause Analysis methodologies can be Rule based, Case based or Probabilistic [1]. In Rule based RCA, a set of rules which map the symptoms to root causes are built. The root cause of an incident is determined by matching the symptom with rule. When necessary conditions of a particular rule are satisfied, the conclusion is obtained. The rule is then triggered. In Case based RCA, a number of cases and their solutions are maintained in a repository. When a new case occurs, the case which has closest similarity to the new case is fetched. Based on the solution of the case retrieved, a solution for the new case is proposed. The correctness of the new solution is validated and then it is decided whether to add this new knowledge to the system. Probabilistic RCA makes use of fault models based on causal graph. These graphs depict the cause and effect relations among system components.

IV. **IMPLEMENTATION APPROACH FOR THE PROPOSED TOOL**

In this section we propose a tool to automate the defect cause analysis of software defects. The tool will make use of historical defect data to perform root cause analysis. Each organization has its own ways of classifying defects. The rules needed to determine the causes of defects are also unique to each organization. The tool is generic and will enable an administrator to configure the rules for defect classification and defect cause identification. The administrator can manually enter the class names and the rules that determine each class. The administrator can also configure the set of attributes that determine each defect cause.

The stages of the proposed tool are:

- Storage and Processing of defect datasets using Hadoop File System and MapReduce
- Classification of Defects using Decision Tree Algorithm
- Fault Tree Analysis using Decision Tree Algorithm

Figure1 shows the overall flow diagram of the proposed tool. The distributed file system of Hadoop will be used to store the defect data, product and project metrics needed for the classification and root cause identification. Two Decision tree based classifiers are used in proposed model. Defect classifier is responsible for classifying the defects and the Fault classifier is

responsible for fault tree analysis of the defect cause. The proposed tool has two phases: a training phase and an operational phase.

Training Phase: During the training phase of the Defect classifier, a training set, where class of each defect is known, is provided as input to the classifier. The Defect classifier creates a classification model based on the training set. Similarly in the case of Fault classifier, the training set is given as input to the Fault classifier. The root cause of each defect in the training phase is known. A model is created using the information in the training set.

Operational Phase: During the operational phase of the tool, new defects are fed to the tool. The Defect classifier classifies the defects based on the model generated during training phase. The defects are classified and displayed to the analysts as dashboard. For e.g. the dashboard may display the defects classified as Blocker, Critical, Major, Minor, Trivial. When an analyst clicks on a particular class of defects, the defects belonging to that particular class are given to the Fault Classifier. The Fault Classifier then identifies the cause of each defect based on the fault tree generated during training phase of the Fault Classifier. A root cause report is generated as output of the operational phase.

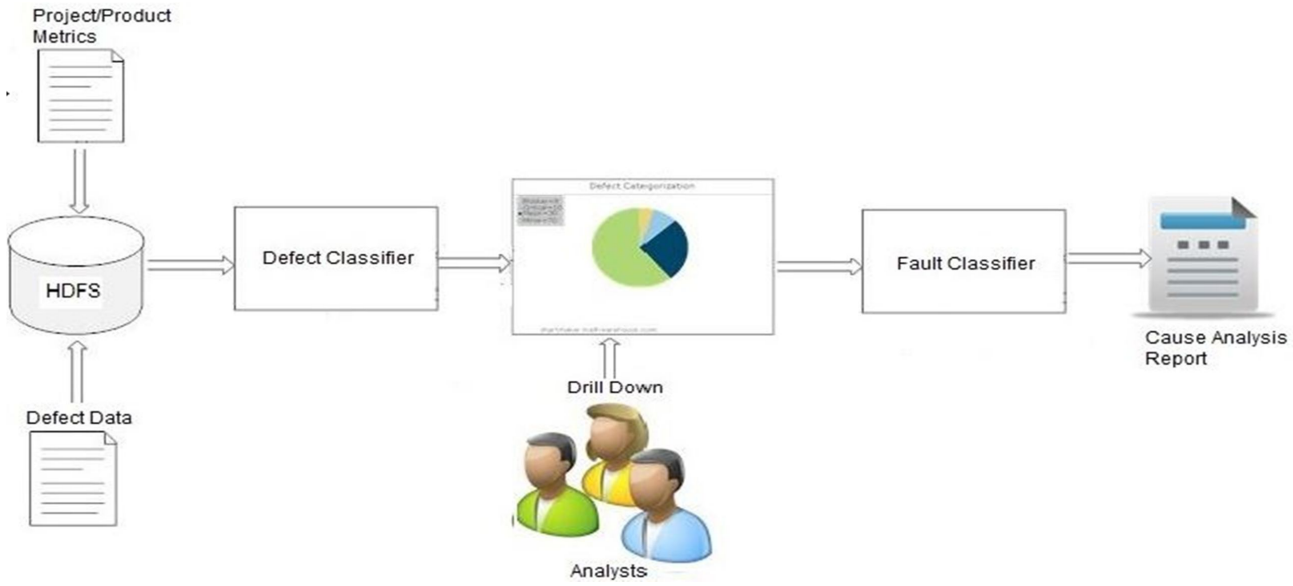

Fig.1 Flow Diagram of the Proposed Tool

A. Storage and Processing of Defect Datasets using Hadoop File System and MapReduce

Hadoop is an open source Java based software framework that supports storage and processing of large datasets in a distributed environment. Hadoop is an open source technology that offers a cheaper option than traditional data warehousing technologies. Hadoop enables applications to be run on a system with multiple nodes involving thousands of terabytes of data. The two major components of Hadoop are

- Hadoop File System (HDFS) for distributed data storage
- MapReduce for parallel processing

Hadoop File System (HDFS) provides a rapid data transfer rate among the nodes and allows the system to continue its operation in the event of a node failure. HDFS has the ability to handle diverse data types from disparate systems. In HDFS, files are stored in a redundant manner across multiple machines ensuring fault tolerance and availability. We are proposing a technology based on HDFS for primarily two reasons. The first reason is that many enterprises have already adopted Hadoop to manage the fast expanding volumes of unstructured, semi structured and structured data. The second reason is that Hadoop is schema less and can handle structured and unstructured data from any number of sources.

MapReduce is a software framework that enables processing of large datasets in parallel. Using MapReduce the computation is done in two stages: the map stage and the reduce stage. Within map stage and reduce stage, the processing is done in parallel. The map and reduce stages operate sequentially such that the reduce stage starts after the map stage. In the map stage a master

node takes the input, divides it into sub problems and assigns the work to worker nodes. Worker nodes process the data and pass the result back to the master node. During reduce operation, the master collects answers to all the sub problems and combines them into an answer to the main problem.

The details of defects reported, product metrics (KLOC, cyclomatic complexity etc.), project metrics (skill set of resources, project implementation language, compiler version, system configuration, project schedule details, slippage details etc.) are stored in HDFS. Decision Tree Algorithm can be implemented in MapReduce to process the data.

B. Defect Classification and Fault Tree Analysis using Decision Tree Algorithm

The proposed tool uses ID3 (Iterative Dichotomiser 3) decision tree algorithm to classify defects and also to create the fault tree for identifying root causes. A decision tree is a flow chart like structure where each non-leaf node represents a test condition on an attribute [14]. Each leaf node represents a class to which the defect belongs. A branch of the tree represents a set of conditions or rules which have to be satisfied by a data item to belong to a class. Decision Tree Algorithm breaks down a dataset into smaller datasets and at the same time incrementally develops the decision tree. The tree is built top down from the root and involves the division of data into subsets that contain instances with similar values. There are a number of decision tree algorithms like ID3, C4.5, CART etc. There are two phases in the implementation of decision trees. The phases are Training phase and Testing phase. During training phase, dataset with classes already known is provided to the decision tree classifier. The classifier develops a model based on the training set. The model is then used during testing phase to classify unknown data.

Proposed tool is recommended to use ID3 decision tree algorithm for classifying the defects. Every organization has its own defect classifications. The rules that determine the classes of defects are unique to each organization. To enable this flexibility, the proposed tool will allow an administrator to specify the classes of defects and the attributes that determine each class. During the training phase, defect data with known classes and rules specified by the administrator are provided to the classifier. Classifier develops a model and uses this model to predict the classes of new defects.

Once the defects have been classified, root cause analysis of the defects is taken up. Fault tree analysis is a deductive procedure to determine the specific cause of failures. The analysis usually begins with a conclusion and then attempts to find the causes of the conclusion by constructing a logical diagram called a fault tree. The proposed tool creates the fault tree for defect cause identification using ID3 algorithm. The conditions which determine the root causes are different for different organizations. The proposed tool accepts the root causes and the attributes for each root cause from the administrator. The fault tree is generated based on ID3 algorithm using the supplied information. The root causes provided by the administrator form the leaf nodes of the fault tree. The rules determine the branch of the tree. The root cause of each defect in the training set is already known in advance. This information, along with the attributes specified by the administrator is used to train the ID3 based fault tree. This knowledge is then used to determine the root cause of new defects.

V. CONCLUSIONS

It is impossible to completely avoid software defects. It is vital to understand the root cause of a defect before fixing it. Failure to perform a root cause analysis and take necessary corrective actions to avoid the particular type of defect, cause the same type of defects to repeat again and again. Root cause analysis can in turn improve the overall reliability and quality of the software development process. Software defect cause analysis paves way for defect prevention and prediction. Higher product quality and enhanced productivity through reduction in rework are the clear outcomes of defect prevention. The tool proposed in this study will help automate the defect cause analysis process of software industries.

REFERENCES

[1] Grace A. Lewis (2010) MESOA 2010 Summary. [Online]. Available: http://www.sei.cmu.edu/workshops/mesoa/2010/MESOA2010-Summary-Lewis.pdf.

[2] M. Surendra Naidu, Dr. N. Geethanjali, "Classification of Defects in Software using Decision Tree Algorithm", *International Journal of Engineering Science and Technology (IJEST)*, vol.05, no. 06,June 2013.

[3] Sakthi Kumaresh and R Baskaran, "Software Defect Prevention through Orthogonal Defect Classification," *International Journal of Computers and Technology*, vol. 11, no.3,pp. 2393–2400, Oct. 2013.

[4] Pranayanath Reddy Anantula and Raghuram Chamarthi, "Defect Prediction and Analysis Using ODC", *International Journal of Computer Science and Information Technologies*, vol. 2(5), 2011, p 2242-2245.

[5] Marcos Kalinowski, David N. Card and Guilherme H. Travassos, "*Evidence-Based Guidelines to Defect Causal Analysis*", IEEE Transactions on Software Engineering, 2012.

[6] David N. Card, "Myths and Strategies of Defect Causal Analysis", Proceedings of Pacific Northwest Software Quality Conference, October 2006.

[7] Petrosky.H, "To *Engineer is Human; the Role of Failure in Successful Design*".

[8] Sakthi Kumaresh and R Baskaran, "Defect Analysis and Prevention for Software Process Quality Improvement", *International Journal of Computer Applications*, Vol. 8, No.7, October 2010.

[9] IEEE Std. 1044-2009: IEEE Standard Classifications for Software Anomalies.

[10] *Quality Standards Defect Measurement Manual*, United Kingdom. Software Metrics Association (UKSMA), October 2000

[11] Stefan Wagner, "*Orthogonal Defect Classification and Defect Types Revisited*", Technische Universität München,Boltzmannstr. 3,85748 Garching b. München, Germany.

[12] Rodriguez, Daniel, Roberto Ruiz, Jose C. Riquelme, and Rachel Harrison, "A Study of Subgroup Discovery Approaches for Defect Prediction," *Information and Software Technology*, 2013.

[13] Tao WANG, Weihua LI, Haobin SHI, Zun LIU, "Software Defect Prediction Based on Classifiers Ensemble," *Information and Computational Science*, vol. 8, pp. 4241-4254, Dec. 2011.

[14] Bhaskar N. Patel, Satish G. Prajapati and Dr. Kamaljit I. Lakhtaria, "Efficient Classification of Data Using Decision Tree," *Bonfring International Journal of Data Mining*, vol. 2, No.1, March 2012.

