Divanshi Priyadarshni Wangoo et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 5, Issue 1, March 2018, pp. 19-23

A Classification based Predictive Cost Model for Measuring Reusability Level of Open Source Software

Divanshi Priyadarshni Wangoo[1] and Archana Singh[2]

[1]Amity University, Noida, Uttar Pradesh, India
[2]Amity University, Noida, Uttar Pradesh, India

Abstract. Measuring the reusability level of open source software's is essential for meeting the overall goals of the reuse business. Mining Software repositories are a great source of knowledge for significant research in the fields of software reusability, software reliability, software bug prediction and removal, change prediction etc. To measure the reusability of a software component requires choosing the right measurement goals which would enhance the predictability of a component or application for reuse. This paper present an architectural model for prediction of component reusability with the right measurement goals chosen for reusability level. The model is based on classification by decision tree induction that will enhance the cost, quality, time and reuse level of a component for software reuse process. The proposed model will consider the code and non-code components to determine the size and amounts of reuse usage level.

Keywords: Software Reuse, Data Mining, Reuse Level, Reuse-Driven Software Engineering Business (RSEB).

I. INTRODUCTION

Software reuse is one of the key parts of the software engineering strategy for improving the performance of the software development process. The Reuse-driven Software Engineering Business (RSEB) includes various dimensions namely, business orientation, engineering orientation, technical sequence and business process reengineering [1]. The dimension of the technical sequence on RSEB requires component based development models for object-oriented programming applications to allow reusing of code which is useful for maintenance, repairing of software defects, testing of the software, documentation and artifacts available in case of open source projects etc. The reusability level of an organization depends upon the components and application systems that are incorporated into a systematic reuse business cycle. There is a need for Object-Oriented Business Engineering techniques that would provide a systematic approach to improve the reusability level of an organization in a reuse business environment. For effective management of reuse business in an organization, there is a need for determining the size and amounts of reuse in non-code components along with the code components [1]. The non-code components account for all those components that are the work products of the stages of the software development life cycle model like use cases, design models or other work products like documents and artifacts associated with a software component and application. Washizaki et al., [2] have proposed a metrics suite for measuring the reusability of black-box components for the development with component reuse which is based on the limited static information that can be obtained from the outside of components without any source codes, to identify the best components in terms of their reusability [2]. B V Prakash et al., [3] have stated a need for new technologies and techniques that are required to reuse the already existing knowledge from software historical data and software repositories such as code bases, execution traces, historical code changes that would help in increasing the reuse level for open source projects [3]. This paper presents a classification based predictive cost model that would measure the reusability of the components available in the open source projects. The reusability is measured by estimating the size and cost of the reusable model. The proposed model will consider both the code and non-code components that are considered for measuring the cost of reusability and will also consider the time and effort spend on the task. For empirical validation of the proposed model the total size of the reuse components in terms of lines of code are taken from the projects of the open source code repositories and are analyzed with data mining tools.

II. MINING OF OPEN SOURCE CODE FOR REUSABILITY

The open source code data is taken from the software repositories that will be mined for measuring their reusability level and estimating their costs along with the time and effort spent in the task. There are many source code hosts like sourceforge, github etc that are used for mining the open source code. Software reuse is the use of existing software or software knowledge to construct new software that can be used as reusable assets in the form of either software products or software knowledge [4]. The components available for reuse can be assembled to analyze the component modules and seamlessly integrate them into industrial information systems to meet various application demands [5]. Jangic et al. [6] have described a large and unabridged data-set of source code gathered and shared as part of the Merobase Component Finder project of the Software-Engineering Group at the University of Mannheim which consists of the complete index used to drive the search engine and the vast majority of the source code modules are accessible through it, also it consists of a tool that enables researchers to efficiently browse the collected data [6]. Data from such repositories are taken to evaluate the reusability of software modules and are a great source for predicting their reuse level.

There are important problems existing in the business-related reuse research that has helped in identification of organizational structure to support corporate reuse programs, staged process models for reuse adoption, and models for estimating return on investment from a reuse program [4]. There are several authors that have modified the cost models such as COCOMO that are used to estimate time and effort for the development of components and of the applications using the components for reusability [1]. As stated by Anas et al., [7] potential components are mined based on the analysis of a set of similar software, some of them may be similar and those providing mostly the same functionalities and differing in few ones. For considering the potentiality of reusable and non-reusable components the same class of data must be analyzed for selecting the potential candidates for reuse. For measuring the code quality, the following attributes can be taken into consideration - functionality, reliability, usability, efficiency, maintain-ability and portability [8]. For measuring the reusability level of a component all the factors can be considered for seeking the quality of reusable level achieved.

III. PREDICTIVE MODEL FOR REUSABILITY BASED ON CLASSIFICATION

The proposed model is named as CRePT: A Classification based Reusability Predictive Tree which is based on the classification by decision tree induction algorithm that will help in predicting the reusability level of the open source projects. Decision tree induction is the learning of decision trees from class-labeled training tuples in which there is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test and each leaf node holds a class label [9]. As data mining is a process of knowledge discovery from large amounts of data, the process usually involves a series of steps. Knowledge discovery as a process consists an iterative sequence of the following gradations- data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation and knowledge presentation [9].

Various models have been proposed for reusability of components and applications. Shatwani et al. [7] have mentioned the greatest obstacle to the wide implementation of CBSE (Component based Software Engineering) as lack of component libraries and have proposed an approach to mine reusable components from a set of similar object-oriented software developed in the same domain by the same developers [7]. Various CBSE (Component based Software Engineering) models have been described in history like Common Object Request Broker Architecture (CORBA), CORBA Component Model (CCM), Sun's Enterprise JavaBeans (EJB), and Microsoft's Component Object Model (COM+) etc. and numerous domain engineering approaches have been reported in various publications such as Domain Analysis and Reuse Environment (DARE), Family-Oriented Abstraction, Specification, and Translation (FAST), Feature-Oriented Reuse Method (FORM), Product Line UML-Based Software Engineering (PLUS) etc. for reusability models and approaches respectively [4]. The proposed model in this paper is not associated with domain engineering but will be using the KNN classification technique for retrieval of software components and enhancing their reusability level. It will also classify the components as potentially reusable and non-reusable. The proposed algorithm can be compared with the ROMANTIC approach proposed by Anas et al., [7] which considers the mining process of the reusable components from a set of similar software only. However, the practical reuse approaches consider potential reuse candidates from a wide variety of domain that combine the matching components existing in different domains and map it as a as a single component candidate for reuse in the required applications. The model proposed in this paper will take the components from such specific domains and can classify the potential candidates for the reusability process. Also, it will take both the code and the non-code part of the component systems for determining their reusability level which is not significantly considered in the previous researches of the software reusability. This model will act as a base to all the available component reusable models and will help the developers to estimate the reuse level and reuse costs of large component systems. This will identify the potential candidates for large-scale reusability purposes. The model will use the KNN classifier as an input to the CReDT algorithm that will generate the decision tree to differentiate between reusable and non-reusable components. The membership to the class is predicted based on the KNN algorithm. The dataset is selected based on the defect tracking status of various open source projects. The implementation and analysis is described in section 4.

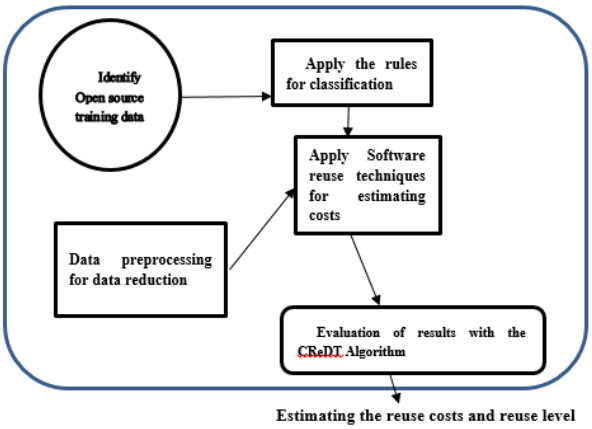

Fig. 1. Proposed model CRePT: A Classification based Reusability Predictive Tree

The proposed model named CRePT which is a Classification based Reusability Predictive Tree model described in Fig.1 above consists of the series of steps starting from identifying the open source training data to evaluating the reuse costs and reuse levels. The purpose of determining the reusability level is to find out potentiality of the components to be reused in the applications with the changing technological advancements.

K-Nearest Neighbor Classification for Retrieval of Software Components

The software components to be reused are retrieved using the K-NN algorithm. The defect tracking dataset used in the K-NN is taken from GitHub. The Eclipse and Mozilla Defect Tracking Dataset contains the bug reported for 4 popular products

retrieved from both Eclipse and Mozilla. [10]. Two similar datasets are used for analysis and includes only selected products information as per the data available from the site. The defect tracking dataset 1 and 2 have 4 examples and 3 attributes. The model is visualized by analyzing the datasets in RapidMiner studio. It can also be used for building knowledge databases for test types to be used for reusing in software projects [11].

The KNN (K-Nearest Neighbor) is a classification technique which is based on learning by analogy i.e., it compares a given test tuple with training tuples that are like it and are described by *n* attributes. Each tuple represents a point in an *n*-dimensional space. When given an unknown tuple, a ***k***-nearest-neighbor classifier searches the pattern space for the *k* training tuples that are closest to the unknown tuple and these *k* training tuples are the *k* - nearest neighbors of the unknown tuple [9]. The software components that are used in the *CReDT* algorithm are retrieved using the KNN classification technique. To measure the closeness of the component with the available component training tuples, the Euclidian distance between two points or tuples is used which is described as follows [9].

For two component training tuples-

$\boldsymbol{C}1 = (c11, c12, : : : , c1n)$ and $\boldsymbol{C}2 = (c21, c22, : : : , c2n)$

The Euclidian distance is given by

$$di(C1, C2) = \sqrt{\sum((C1i - C2i)2)} \quad (1)$$

The above equation will measure the relative closeness of the components in terms of distant metric of their relatedness. The results and analysis are described in section 4. The next subsection gives a brief overview of the *CReDT* algorithm used for determining the reusability of the open source data.

The standard deviation can be used to measure the variability of the potential software components for reusability and the calculation of associated variance is defined in equation 2 as below:

$$R^2 = \sum i = 1 \text{ to } n \frac{(Nri - Nri')2}{n-1} \quad (2)$$

Where R^2 is the variance for measuring the variability of the reuse components, *Nri* are the potential components retrieved from the *CReDT* algorithm and *Nri'* is their standard mean. The standard deviation R of the potential reusable components will be calculated by taking the square root of the variance defined in equation 3 as follows:

$$R = \sqrt{R^2} \quad (3)$$

CReDT: A Classification by Decision Tree Induction Algorithm for Determining the Reusability of the Open Source Data

The proposed algorithm for measuring the reusability level of an open source software is named as *CReDT* which is a classification by decision tree induction algorithm that takes the training tuples of data which consists of the two types of components- components with reuse and components with no-reuse. In open source software projects not all the component modules are taken for the reusability into the new application systems therefore the training data will include both the components and output will be a decision tree that will help in identifying the potential of the component to be reused into a new application system.

CReDT Algorithm: **To generate a decision tree predicting reusability to determine the potential candidates for reuse**

CReDT Algorithm: **To generate a decision tree predicting reusability to determine the potential candidates for reuse**

Input: Partitioning of the data, P, to differentiate between reusable and non-reusable components using KNN classification

Output: A decision tree predicting reusability

Method:

Step 1- Create a node N

Step 2- If the input training data consists of components of the same class, partition them into reusable and non- reusable components N_r and N_{nr} respectively

Step 3- If N_r belongs to the class of reusable components then

Step 4- Include N_r as a potential candidate for reuse

Step 5- Otherwise consider the next node in the decision tree

Estimating the Reuse Costs and Reuse Level in the Proposed Model

The cost of reuse and reuse level can be determined by considering the reusable and non-reusable component modules. As open source software projects consist of numerous component modules from which some modules may be required for reusability in the application systems, the cost incurred with them is difficult to estimate with the availability of non-reusable modules in the same project. Furthermore, with time changing the reusability level of the open source software changes with new technologies. Therefore, a simplified and effective reuse cost and effort model for reuse measurement as mentioned below will be used for the estimating the reuse cost and reuse level. It will be considering the cost of no reuse of the components in the application systems at a time and measures the software size in function points or line of code. The formulas defined below can be used as an enhancement to the formulas

defined by Jacobson et al., in [1] which does not take the time as an essential factor for measuring the reusability level of the components. Time is considered as an important factor for deciding the reusability level as per the technological advancements.

C (T) $_{non\text{-}reusable}$ = The cost of developing an application system without reusing the component system at a given time T

C (T) $_{reusable}$ = The cost of developing an application system with reusing the component system at a given time T

Now we will use the formulae for measuring the reuse level from some set of component systems

$$\text{Reuse level, } RL = \frac{Total\ size\ of\ reuse\ components\ used\ in\ the\ application\ systems}{Size\ of\ application\ components} \quad (4)$$

D $_{reusable}$ = The cost involved in reusing a component

The overall cost of developing an application with reuse usually has two parts – one is the (1-RL) part which is developed without reuse at normal cost and the other part is the RL part which is developed with reuse at a lower cost as defined by Jacobson et al. in [1].

The estimation of the costs at a particular time T can be done as follows-

$$C(T)_{component\ part\ with\text{-}\ reuse} = C(T)_{reusable} * (RL * D_{reusable}) \quad (5)$$

$$C(T)_{component\ part\ with\ -\ no\text{-}reuse} = C(T)_{non\text{-}reusable} * (1\text{-}RL) \quad (6)$$

Therefore the total cost at a particular time T with reuse becomes

$$C(T)_{total\text{-}with\text{-}reuse} = C(T)_{component\ part\ with\text{-}reuse} + C(T)_{component\ part\ with\ -\ no\text{-}reuse} \quad (7)$$

$$C(T)_{non\text{-}reusable} = C_{non\text{-}reusable} * (RL * D_{use} + (1\text{-}RL)) \quad (8)$$

At a given time T.

Thus, the reuse level of an open source software can be determined by including both the reusable components and non-reusable components.

From the above equations we can deduce the amount of reuse level in the components taken as given below-

$$RL_{amount} = \frac{RL*(1-RL)}{Total\ number\ of\ reusable\ components} \quad (9)$$

The above equation from (5) – (9) can be used for measuring the reusability level of the open source data projects. This will make the task of the software developer easy and time managed. Also, the cost of the software development can be reduced along with the enhancement in the quality of the software developed in the organization employing Reuse-Driven Software Engineering Business (RESB) practices.

IV. METHODOLOGY USED AND ANALYSIS

The open source data is mined using the classification algorithm for inputs in the training sets where the data sets are taken relatively small as compared to large amounts of data sets taken for the mining process. The algorithm *CReDT* can be applied to datasets of the open source software for evaluating their reusability levels at a particular time. The algorithm designed will enhance the predictive reusability of the data set taken for the specified components in Reuse-Driven Software Engineering (RESB). Further it will improve the quality of the components selected for reusability in open source software projects.

For implementing the KNN classification technique described in section 3.1 above, the datasets taken from github are used as described in Table 1 and Table 2 below [10]. These datasets are subjected to the KNN classification technique in RapidMiner and the results of the model generation are displayed in Fig. 1.

Table 1. Defect tracking dataset 1

Product	Number of components	Number of reports
Platform	22	24.775
JDT	6	10.814
CDT	20	5.64
GEF	5	5.655

Table 2. Defect tracking dataset 2

Product	Number of components	Number of reports
Core	137	74.292
Firefox	47	69.879
Thunderbird	23	19.237
Bugzilla	21	4.616

The reusability level of the defect tracking dataset 1 as per the equations (5) to (9) is calculated as below-

Reuse level, $RL = \frac{Total\ size\ of\ reuse\ components\ used\ in\ the\ application\ systems}{Size\ of\ application\ components}$

So, the reusability level of the defect tracking dataset 1 can be defined as below (here the size of the reusable components as defined in terms of the number of the components used for all the product and the size of the application components are taken arbitrary, for this dataset for illustration purpose)

RL= 22+6+20+5 / 75 = 70.66%

Similarly, the reusability level of the defect tracking dataset 2 as per the equations (5) to (9) is calculated as below-

RL= 137+47+23+21 / 275 = 82.9%

Therefore the defect tracking dataset 2 has more reusability level that the defect tracking dataset 1 as the number of reusable components are more.

The following figure shows the implementation of the KNN classifier on the two defect tracking datasets in the RapidMiner studio. The result are based on 1-KNN where the no of defect reports is taken as the special attribute that will predict the retrieval and classification of the components into reusable and non-reusable components. The model generation results are depicted in the figure below-

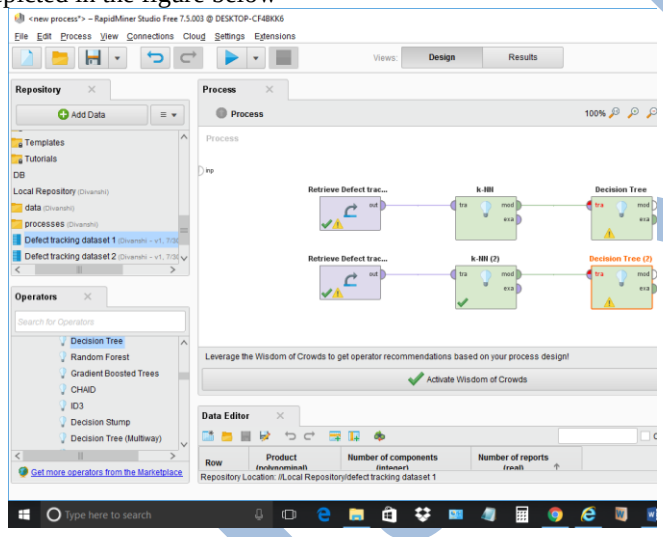

Fig. 1. The Model for KNN classification on the defect tracking dataset 1 and 2 using RapidMiner Studio

V. CONCLUSION AND FUTURE WORK

This paper present an architectural model for prediction of component reusability with the right measurement goals chosen for reusability level. The goals for the measurement are based on determining the reuse level and reuse cost that considers both the reuse and non-reuse components. The model is based on classification by decision tree induction that will enhance the cost, quality and time and reuse level of a component for software reuse process. The proposed model will consider the code and non-code components to determine the size and amounts of reuse usage level. The future work includes the application of the algorithm to huge datasets of open source software for precise estimation of reusability level and costs and incorporation of the measurement analysis of the the reusability level of the components in the data mining software's.

References

[1]. Jacobson, I., M. Griss, and P. Johnson. "Software Reuse: Architecture, Process and organization for Business Success–Addison Wesley." *Reading, MA, May* (1997).

[2]. Washizaki, Hironori, Hirokazu Yamamoto, and Yoshiaki Fukazawa. "A metrics suite for measuring reusability of software components." *Software Metrics Symposium, 2003. Proceedings. Ninth International*. IEEE, 2003.

[3]. Prakash, BV Ajay, D. V. Ashoka, and VN Manjunath Aradhya. "Application of data mining techniques for software reuse process." *Procedia Technology* 4 (2012): 384-389.

[4]. Frakes, William B., and Kyo Kang. "Software reuse research: Status and future." *IEEE transactions on Software Engineering* 31.7 (2005): 529-536.

[5]. Frakes, William B., and Kyo Kang. "Software reuse research: Status and future." *IEEE transactions on Software Engineering* 31.7 (2005): 529-536.

[6]. Janjic, Werner, et al. "An unabridged source code dataset for research in software reuse." *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013.

[7]. Shatnawi, Anas, and Abdelhak-Djamel Seriai. "Mining reusable software components from object-oriented source code of a set of similar software." *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*. IEEE, 2013.

[8]. Touw, Egbert. "Multi-faceted Reliability Assessment Techniques: An Industrial Case Study." *Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on*. IEEE, 2017.

[9]. Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[10]. https://github.com/ansymo/msr2013-bug_dataset

[11]. Uetsuki, Keiji, and Mitsuru Yamamoto. "Improvement of Description for Reusable Test Type by Using Test Frame." *Software Testing, Verification and Validation Workshops (ICSTW), 2017 IEEE International Conference on*. IEEE, 2017.

