Prediction of Success or Failure of Software Projects based on Reusability Metrics using Support Vector Machine

R. Sathya
Assistant professor, Department of Computer Science & Engineering
Annamalai University
Annamalai Nagar, India.
sathya_vai@yahoo.com

Dr. P. Sudhakar
Assistant professor, Department of Computer Science & Engineering
Annamalai University
Annamalai Nagar, India.
kar.sudha@gmail.com

Abstract— In the field of computer science & engineering and software industry the term reusability means usage of existing software assets or previously developed code in the software development process. The assets of the software are products and by-products of the product development life cycle which includes code, test cases, software designs and code documentation. The process of modifying the existing assets as per the need and specific requirements is called leveraging. But the reusability process creates a new version of the existing assets. So always reusability is preferred rather than leveraging. To identify the quality of the reusability of the software components various software metrics are available. But the framework or model that can predict the reusability of the software assets are needed to be developed. The reusability metrics must be identified during the design or coding phase and that can be used to reduce the rework needed develop a similar software module. This can much improve the productivity due to the probabilistic increase in the reuse level. In this study various software metrics representing the software reusability nature of the software components are collected in relation with a particular software project to form a database. The database is divided in to training and test set and Support Vector Machine is trained using the Radial Basis Function (RBF) to predict whether the software component can be reused or not.

keywords: *Support Vector Machine (SVM), Radial Basis Function (RBF), software reusability, software metrics.*

*****

I. INTRODUCTION

Software reusability is the process of developing or upgrading software components using existing or previously developed software components [1]. A good software reusability approach increases the productivity, reliability and quality of the software being developed. In Parallel it also reduces the overall development cost and time. In the initial level an investment is needed to form a repository of reusable software components. In other words, the development of reusable components improves the quality, minimizes the amount of work needed for developing future components and thus ultimately reduces the risk of developing new components which depends on the reusable software repository.

Among the various software components, the simplest form of reusability is exhibited by subroutines or functions. A part of code is organizes into layers using modules or namespaces. The objects of the object oriented programming offer an advanced form of reusability. But it is highly difficult to measure the level or the score of reusability. The capacity to build larger software modules from smaller components and to identify the similarities among those components mainly decides the reusability. Reusability is an essential characteristic of a software component. It requires compulsory management of building, packaging, configuration, installation, deployment and maintenance issues and if these are not managed properly the software may appear to be reusable from the design point of view but practically the software components cannot be reused. Reuse is a process of conceiving solution to a problem using the solutions already available for the sub-problems.

The process of reuse can be completed by following the above six major steps [2]. In each step preparation for the next phase is carried out. Initially a reuse plan is developed by studying the given problem and available solutions to it, after that a solution structure is identified. Next the structure of the solution is reconfigured to improve the reusability in the upcoming phases. Then the existing reusable components are acquired, instantiated and/or modified. The reused software components are then integrated together with the newly developed components and finally the complete product is validated and evaluated.
The software reusability has a several benefits [3] and they are summarized below.

The reusable software components are tested and tried in already working systems and when they are used to build new software module the reliability is increased. During the initial usage of the software components the bugs in the design and implementation are identified and removed. So when reused the chance of failures are greatly reduced. When the software components are reused it reduces the uncertainty in the cost estimation of the software. This is an essential factor as it reduces the error in the overall project cost estimation which is particularly true when majority of the software components are built from reusable components. It reduces the requirement of coding specialist doing the same work on various software projects. Instead the specialist can develop reusable software components that can be used whenever similar problem arises during the software development life cycle. Some standard or conventional software components can be made reusable so that there will not be change in the functioning or user interface of them. To mention few example, the menus in a user interface can be developed and maintained as a reusable software component so that a common menu format can be presented to the user. Building and introducing the system to the market as early as possible is more important rather the reducing the overall development cost. The software reusability can increase the overall production speed as both the development and validation time required for most of the component is greatly reduced.

The IT professionals have started realizing the importance of software reusability to manage the situation called as software crisis [4 -8]. As discussed previously software reuse increase the productivity and quality [9-10] of the software product. The concept of software reuse introduced in 1968 which introduced new scope for software design and development [11]. The software development people are slowly drifting towards the software reusability by which new software system can be built using the existing systems without much effort and time. As the software reuse poses many benefits there has been much discussion on the topic of reusability including the research directions for software reusability. The software developers are following many of the software reuse approaches in software designs, patterns, and templates and also in searching, matching and modeling tools. The entire software industry is moving towards large scale software re-usage. So at this juncture it is essential to predict or identify whether a software component is reusable or not while it is being developed and maintained.

II. DATASET

For an IT concern which has not implemented software reuse concepts, there exists a chance to develop reusable software components from the scratch or they can identify reusable components from their software repository. The prediction or identification of reusability of the software components present in the repository is accomplished by using a set of software metrics and a mathematical model like SVM or Artificial Neural Network (ANN). This work is focused on building such a mathematical model to predict the reusability of the software. In this work PROMISE Software engineering repository data is used which contains 29 attributes describing the reusability of the software [12]. Among the various software engineering research datasets, the PROMISE repository is a specialized research dataset repository. The majority of the attributes are of categorical type. So it is essential to encode the categorical features using one-of-K coding scheme before giving them as input to the SVM for training.

III. SOFTWARE REUSABILITY METRICS

The metrics used in this work are derived based on the answers given by industry people for a set of questionnaire. For each of the software project data points were coded in to either categorical or numerical data and can be divided in to two categories namely state metrics and control metrics. The state metrics composed of attributes for which the company has no control such as the size and application domain. The control metrics composed of attributes such as type of reusability approach followed, modification carried out to the existing development cycle, commitments from the management. The list of the metrics is given in Table 1.

Table 1. List of Metrics representing software reusability

S. No.	Name of the Attribute	Remarks
1	Project ID	Project identifier
2	Software Staff	Number of staffs involved in software development
3	Overall Staff	Total No. of staffs involved in the software project
4	Software Production	Represents the similarity of the software with other
5	Software and Product	Represents either the software is embedded in a product or process
6	SP maturity	Represents certification or standard of the S/W
7	Application Domain	Represents the domain of the Software product
8	Type of Software	Represents the nature of the software i.e. real time, embedded, or other.
9	Size of base line	The size of software module on which reusability was imposed.
10	Development approach	Represents the analysis and design approach used.
11	Staff Experience	Represents the experience of the staff in code development
12	Reuse approach	Represents the coupling between the reusable and other components of the software
13	Work-products	Represents the type of components reused.
14	Domain Analysis	Tells whether or not domain analysis were performed
15	Origin	Represents whether components are developed from scratch or re-engineered.
16	Independent team	Tells whether development and

		reuse team are same or not.
17	When assets developed	Tells when the reusable components were developed
18	Qualification	Tells whether reusable components undergo a qualification process or not.
19	Configuration management	Tells whether the reusable components have a configuration management and change control or not.
20	Rewards policy	Tells whether there is reward to staff if reusability is adopted.
21	Assets	Number of reusable assets in the software repository.

IV.CLASSIFIERS FOR PREDICTION OF SUCCESS OR FAILURE OF SOFTWARE REUSABILITY

The prediction of success or failure is a kind of binary classification problem. The three classifiers used in this work are Support Vector Machine, Bayesian Networks, and Gaussian Mixture Model. All three classifiers don't have built-in feature selection ability and are commonly used in the machine learning applications. Support Vector Machine (SVM) is a mathematical model which has a supervised learning algorithm capable of analyzing data and identifying patterns in it. The classification process is done in two steps one is training phase and the other one is testing phase. The labeled feature vectors are fed as input during the training time and the data to be classified is given in the testing phase. The accuracy of classification depends on the efficiency of the trained model. In this work the support vector machine with Gaussian kernel is used and it is denoted by the equation,

$$K(x, y) = exp\left(\frac{-\|x - y\|^2}{\sigma^2}\right) \qquad (1)$$

Bayesian Networks (BNs) are probabilistic graphical models used to represent knowledge regarding an uncertain domain. Each node in the graph denotes a random variable and the edges connecting the nodes denote the dependency among the respective random variables. The dependencies in the graph are calculated using statistical and computational methods. Therefore, the concepts of BNs are combined from graph, probability, computer science and statistical theories. Bayesian Network can be used even when the value of some of the attributes are missing.

The Gaussian Mixture Model (GMM) is a useful supervised learning classification algorithm which can be used for classification of *N*-dimensional dataset. During the training phase GMM is built for each class of data. In this work two GMM has to be constructed to fit the defective and non-defective data samples. There will not be any interactions between GMM of different classes. At the classification phase the unknown-class data is given as input to GMM of each class. The predicted class is the one associated with the GMM with the maximum probability.

V. EXPERIMENTS AND RESULTS

In this training dataset out of 100% records nearly 75% records were belonging to software components that can be reused i.e. the class label for those records were marked as '1' and 25% records belong to non-reusable category. This causes imbalanced data. This problem is crucial in classification or clustering because it is needed to maximize the instances of recognizing the minority class. Two methods for dealing with class imbalances are: oversampling and downsizing. Oversampling consists of re-sampling the small class at random until it contains as many examples as the other class. Downsizing consists of the randomly removed samples from the majority class population until the minority class becomes some specific percentage of the majority class.

Using SMOTE or the Synthetic Minority Over-Sampling Technique the non-reusable components record was made 40%. SMOTE is an oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances.

- Feature Selection

Information theory techniques can be used for feature selection in time series prediction or pattern recognition. These techniques focus on maximizing the mutual information between the input and output data. But this procedures' computational complexity is high as the joint entropy is to be calculated. To calculate joint entropy, the joint probability distribution has to be estimated. To eliminate this computational effort, the feature selection task can be accomplished using the principle of minimum redundancy and maximum relevance. This method maximizes the mutual information between the input and output data indirectly with a low computational cost. But still the combinatorial optimization problem i.e. the check for all possible combination of features requires high computational effort. Because of these high computational costs, a simple and efficient method using incremental search which can produce a quasi-optimal solution was proposed in the previous literatures. With this background, to reduce the computational cost the combinatorial optimization was performed using the Genetic Algorithms. The algorithm output a vector of indices of the features that forms the optimal set of features. Here the order of features has no relation with their importance.

The parameters of the Genetic Algorithm are fixed as follows:

Size of the population	- 100
Max. Number of generation	- 50
Crossover Probability	- 0.7
Mutation Probability	- 0.4
Crossover strategy	- Random single point

In the initial stage the chromosome population is generated randomly. The number of chromosome in the initial population decides the rate of convergence. If the population is larger the problem converges slowly else if the population is small, it converges quickly and explores only a smaller portion of the search space. The optimal feature selected by the genetic algorithm consists of the following attributes namely Software Production, Software and Product, SP maturity, Application Domain, Type of Software, Size of base line, Development approach, Reuse approach, Work-products, Domain Analysis, Origin, when assets developed, Rewards policy, and Assets. Out of 29 attributes (28 attributes and 1 class label) of the dataset 14 attributes were selected which constitute the optimal feature set. The feature set was then used to train the classifiers. In Fig. 2 the fitness function of the genetic algorithm for various generations is presented.

The accuracy of all the classifiers used for the prediction is presented in the Fig. 3. The predictive models constructed using the SVM are computationally more expensive when compare to the GMM and Bayesian Network. As the number of records is limited it is obvious that the SVM outperforms the GMM and Bayesian Networks. If the number of records available is huge the GMM can perform much better as the estimates of the model parameter are better. In the case of SVM the availability of more data will not change the position of the support vectors. The experiments are conducted in this work to exhibit the suitability of the classifiers for a given optimal feature set where the performance of all the classifiers are almost same.

VI. CONCLUSION

This work presents a method to predict the success or failure of a software project based on the software metrics which are related to the reusability of the software components. The projects developed in the various domains of the software industry uses an object oriented or procedural approach for software development and are expected to be successful. But around one–third of the software projects were classified as failed because of lack of reuse processes, and non-modification of non-reuse processes. The main cause for the failure of software components are due to the lack of commitment from the top management people or lack of awareness of the importance of these factors. All the projects that are classified as success implements the concept of reusability in their respective project development. This is reflected from the attributes related the reusability of the software modules. Among the three classifiers used for the prediction the SVM with Gaussian kernel performed better than the other classifiers as the Gaussian kernel perfectly fits for these kind of training data.

REFERENCES

[1] http://en.wikipedia.orglwikilReusability

[2] Kang C. Kyo, Cohen Sholom, Holibaugh Robert, Perry James, Peterson A. Spencer, "A Reuse-Based Software Development Methodology", Application of Revisable Software Components Project Special Report, January 1992.

[3] Somerville Ian, "Software Engineering", 7th edition.

[4] Smith, E., AI-Yasiri, A. and Merabti, M., "A Multi-Tiered Classification Scheme for Component Retrieval", Proceedings 24th Euro Micro Conference-1998, vol. 2, Aug 25-27,1998, pp. 882 -889.

[5] Basili, V.R, "Software Development: A Paradigm for the Future", Proceedings COMPAC'89, Los Alamitos, Calif.: IEEE CS Press, 1989, pp. 471-485.

[6] Basili, V. R and Rombach, H. D., 'The TAME Project: Towards Improvement-Oriented Software Environments", IEEE Trans. Software Eng., vol. 14, no. 6, June 1988, pp. 758-771.

[7] Boehm, B.W. and Ross, R, "Theory-W Software Project Management: Principles and Examples", IEEE Trans. Software Eng., vol. 15, no. 7, 1989, pp. 902.

[8] Boehm, B.w., "A Spiral Model of Software Development and Enhancement", Computer, vol. 21, issue 5, May 1988, pp. 61 -72.

[9] Boehm, B., "Managing Software Productivity and Reuse", Computer, vol. 32, issue 9, Sept 1999, pp. 111 - 113.

[10] Frakes, W.B. and Fox, C.J., "Quality Improvement Using a Software Reuse Failure Modes Model", IEEE Trans. Software Eng., vol. 22, no. 4, Apr 1996, pp. 274-279.

[11] Gomes, P and Bento, C., "A Case Similarity Metric for Software Reuse and Design", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 15, issue 1,2001, pp. 21-35.

[12] *Menzies, T., Krishna, R., Pryor, D. (2016).* The Promise Repository of Empirical Software Engineering Data.

[13] O. Ludwig and U.Nunes; " Novel Maximum-Margin Training Algorithms for Supervised Neural Networks;" IEEE Transactions on Neural Networks, vol.21, issue 6, pp. 972-984, Jun. 2010.

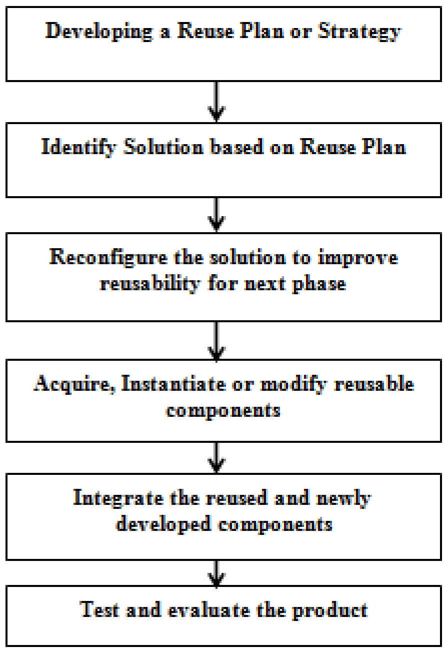
Developing a Reuse Plan or Strategy
Identify Solution based on Reuse Plan
Reconfigure the solution to improve reusability for next phase
Acquire, Instantiate or modify reusable components
Integrate the reused and newly developed components
Test and evaluate the product

Fig. 1 Flow of Reusability process

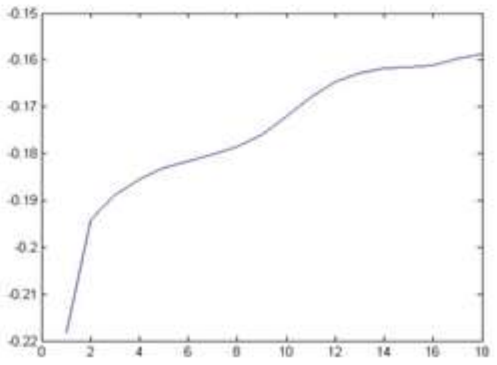

Fig. 2 Evaluation of Fitness function over generations

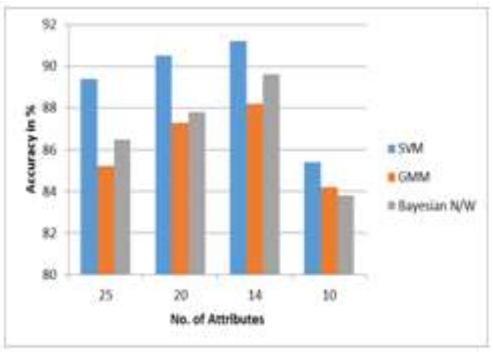

Fig. 3 Performance comparison of classifiers