A SVM Based Approach For Reusability Evaluation of Object Oriented Based Software Components

Prof. Kulwinder S.Mann and Amanpreet Singh

Abstract—Software reuse is the process of implementing or updating software systems using existing software assets. The software industry is moving toward large-scale reuse, resulting in savings of time and money. To develop a new system from scratch is very costly. This has made custom software development very expensive. It is generally assumed that the reuse of existing software will enhance the reliability of a new software application. This concept is almost universally accepted because of the obvious fact that a product will work properly if it has already worked before. SVM is a useful technique for data classification. Even though it’s considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

Keywords—Software Reuse, SVM Clustering, Metrics and Intelligent Systems.

I. INTRODUCTION

SOFTWARE reuse [12] is the process of implementing or updating software systems using existing software assets. Software assets or components include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites. Anything that is produced from a software development effort can potentially be reused.

The reusability is the quality of a piece of software, that enables it to be used again, be it partial, modified or complete.

Software professionals have recognized reuse as a powerful means to potentially overcome the situation called as software crisis [1-3]. Software Reuse promises significant improvements in software productivity and quality. Reusable software components have been promoted in recent years. The software development community is gradually drifting towards the promise of widespread software reuse, in which any new software system can be derived virtually from the existing systems. As a result, an increasing number of organizations are using software not just as all-inclusive applications, as in the past, but also as component parts of larger applications. In this new role, acquired software must integrate with other software functionality.

Software reusability is an attribute that refers to the expected reuse potential of a software component. Software reuse not only improves productivity but also has a positive impact on the quality and maintainability of software products. Given the attractive payoff of reusing software, there have been several efforts undertaken to discuss the topic of reusability, including overviews of software reusability research directions and software reusability in practice. Developers are adopting many of these reuse approaches, including reuse in product lines, design patterns templates, reference architectures and advanced searching, matching, and modeling tools. Many other reuse approaches, such as product lines, design patterns, and context-independent techniques, address reuse in different ways and have also demonstrated benefits.

The software industry is moving toward large-scale reuse, resulting in savings of time and money. To develop a new system from scratch is very costly. This has made custom software development very expensive. It is generally assumed that the reuse of existing software will enhance the reliability of a new software application. This concept is almost universally accepted because of the obvious fact that a product will work properly if it has already worked before [6-11].

A component can be considered an independent replaceable part of the application that provides a clear distinct function. A component can be a coherent package of software that can be independently developed and delivered as a unit, and that offers interfaces by which it can be connected unchanged with other components to compose a larger system.

There are two approaches for reuse of code: develop the code from scratch or identify and extract the reusable code from already developed code. For the organization that has experience in developing software, but has not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed software systems or legacy systems.

Prof. Kulwinder S.Mann is working as Associate Professor, in Department of CSE at GNDEC, Ludhiana, Punjab, INDIA.
Amanpreet Singh is pursuing M.Tech. (CSE) from GNDEC, Ludhiana, Punjab, INDIA(e-mail:er.amandhillon@hotmail.com).
The contribution of metrics to the overall objective of the software quality is very well understood and recognized. But how these metrics collectively determine reusability of a software component is still at its naïve stage.

There are two forms of reuse and they are as:
- Horizontal Reuse.
- Vertical Reuse.

Horizontal reuse refers to software components used across a wide variety of applications. In terms of code assets, this includes the typically envisioned library of components, such as a linked list class, string manipulation routines, or graphical user interface (GUI) functions. Horizontal reuse can also refer to the use of a commercial off-the-shelf (COTS) or third-party application within a larger system, such as an e-mail package or a word processing program. A variety of software libraries and repositories containing this type of code and documentation exist today at various locations on the Internet.

Vertical reuse, significantly untapped by the software community at large, but potentially very useful, has far reaching implications for current and future software development efforts. The basic idea is the reuse of system functional areas, or domains that can be used by a family of systems with similar functionality [5]. The study and application of this idea has spawned another engineering discipline, called domain engineering. Domain engineering is "a comprehensive, iterative, life-cycle process that an organization uses to pursue strategic business objectives. It increases the productivity of application engineering projects through the standardization of a product family and an associated production process." [4] Which brings us to application engineering, the domain engineering counterpart: "Application engineering is the means by which a project creates a product to meet a customer's requirements. The form and structure of the application engineering activity are crafted by domain engineering so that each project working in a business area can leverage common knowledge and assets to deliver a high-quality product, tailored to the needs of its customer, with reduced cost and risk." [4] Domain engineering focuses on the creation and maintenance of reuse repositories of functional areas, while application engineering makes use of those repositories to implement new products.

The paper is organized as follows: section II explains about the methodology followed and section III the result of the study. Finally conclusions of the research are presented in section IV.

II. METHODOLOGY

The aim of Metrics is to predict the quality of the software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. The requirement today is to relate the reusability attributes with the metrics and to find how these metrics collectively determine the reusability of the software component. To achieve both the quality and productivity objectives it is always recommended to go for the software reuse that not only saves the time taken to develop the product from scratch but also delivers the almost error free code, as the code is already tested many times during its earlier reuse.

A great deal of research over the past several years has been devoted to the development of methodologies to create reusable software components and component libraries, where there is an additional cost involved to create a reusable component from scratch[13-24]. That additional cost could be avoided by identifying and extracting reusable components from the already developed large inventory of existing systems. But the issue of how to identify good reusable components from existing systems has remained relatively unexplored. Our approach, for identification and evaluation of reusable software, is based on software models and metrics. With the objective of taking advantage of the features of the SVM technique, in this paper SVM classifier based approach [29] is used to economically determining reusability of software components in existing systems as well as the reusable components that are in the design phase. Inputs to the system are provided in form of five object oriented metric values as representation of the attributes of the software component and output is being obtained in terms of reusability.

Reusability evaluation System for Object Oriented Software Components can be framed using following steps:

I) Selection and refinement of metrics targeting the reusability of Object Oriented software system and perform parsing of the software system to generate the Meta information related to that Software. The metric of the [25-28] are used and the list of metrics is as under:
- Weighted Methods per Class (WMC)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOC)
- Coupling Between Object Classes (CBO)
- Lack of Cohesion in Methods (LCOM)

II) Calculate the metric values of the sampled software components.

III) Use SVM Clustering based prediction system for the Reusability Prediction. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships.

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyper planes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy
variable is created with case values as either 0 or 1. Thus, a
categorical dependent variable consisting of three levels, say
(A, B, C), is represented by a set of three dummy variables:
A: \{1 0 0\}, B: \{0 1 0\}, C: \{0 0 1\}

To construct an optimal hyper plane, SVM employs an
iterative training algorithm, which is used to minimize an
error function.

IV) The performance criterion taken is the classification
Accuracy%. It is the percentage of the predicted values that
match with the expected values of the reusability for the given
data. The best system is that having the high Accuracy value.

III.RESULTS

The proposed Neural based methodology is implemented in
MATLAB 7.4. MATLAB (Matrix Laboratory) environment is
one such facility which lends a high performance language for
technical computing.

The object oriented dataset considered have the output
attribute as Reusability value. The Reusability in the dataset is
expressed in terms of two numeric labels i.e. 1 and 2. The
label 1 represents Non-Reusable and the label 2 represents the
Reusability Label. The statistics of the count of the number of
examples of certain reusability label is shown in the Table I.

The given data is with five Input Attributes i.e. WMC, DIT,
NOC, CBO, LCOM, and one Output attributes named as
Reusability Level of the Software Component. Then SVM
clustering based algorithm is implemented in Matlab 7.4.

First of all, random selection of training and test sets are
made. Thereafter, Training of support vector machine
classifier is performed with the training dataset created. The
training data is provided to SVM in form of Matrix, where
each row corresponds to an observation or replicate, and each
column corresponds to a feature or variable. Groups are also
provided to the SVM as Column vector, character array, or
cell array of strings for classifying data in Training into two
groups. It has the same number of elements as there are rows
in Training. Each element specifies the group to which the
corresponding row in Training belongs.

The trained SVM is now used to classify the test dataset
and the performance of the classification is recorded in terms
of the Correct Rate and Error Rate. The proposed algorithm is
run for 10 times the classification Accuracy is recorded as
shown in table II.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics of the Reusability Output Attribute in the Dataset</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Number of Instances</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy Percentage of the System</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Run</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>1th</td>
</tr>
<tr>
<td>2nd</td>
</tr>
<tr>
<td>3rd</td>
</tr>
<tr>
<td>4th</td>
</tr>
<tr>
<td>5th</td>
</tr>
<tr>
<td>6th</td>
</tr>
<tr>
<td>7th</td>
</tr>
<tr>
<td>8th</td>
</tr>
<tr>
<td>9th</td>
</tr>
<tr>
<td>10th</td>
</tr>
<tr>
<td>Best Value Accuracy%</td>
</tr>
</tbody>
</table>

As evident from the table above the Average Correct Rate
or the Accuracy of the proposed system is 83.54%.

IV. CONCLUSIONS

In this study Support vector machines (SVM) based
classification approach is evaluated for Reusability Prediction
of Object based Software systems. Here, the metric based
approach is used for prediction. Reusability value is expressed
in the two linguistic values. Five Input metrics are used as
Input and the training of Support vector machines is
performed, thereafter performance of the System is recorded
for prediction of reusability of the software modules. As
deduced from the results it is clear that Average Correct Rate
or the Accuracy of the proposed system is 83.54% and the
best Classification Accuracy percentage is 88.3721%, which
is satisfactory enough for the prediction of the Object based
reusable modules from the existing reservoir of software
components. So, the proposed system can be used for
identification and later can be extracted and saved in the reuse
repository for its reuse.

REFERENCES

        Prentice Hall of India.
        Ada Code”, SPC Technical Report, ADA_REUSE_HEURISTICS-
        90011-N, March 1990.
        Progress Report”, SPC Technical Report, SALVAGE ADA
        Improvement Oriented Software Environments”, IEEE Trans.
        Future”, Proceedings COMPAC’89, Los Alamitos, California, IEEE CS
        for Characterizing Reusable Software”, Proceedings of the
        Australian Conference on Software Metrics, Australia, July, 1993, pp.
        234-237.
        Based Approach to the Software Metrics” Proceedings of the 5th
        International Conference on Software Engineering and Knowledge
        Information Abstraction System”, IEEE Trans. on Software Engineering,


