A MODEL SYSTEM FOR EFFECTIVE CLASSIFICATION OF SOFTWARE REUSABLE COMPONENTS

Ramu Vankudothu¹ and P.Shireesha²

¹Research Scholar, Department of Computer Science, Kakatiya University, Warangal, Telangana, India. ramuvankudoth86@gmail.com

²Assistant Professor, Department of Computer Science, Kakatiya Institute of Technology and Science, Kakatiya University, Warangal, Telangana, India. rishapakala@yahoo.co.in

ABSTRACT

Much of the research and development in Component retrieval is aimed at improving the effectiveness and efficiency of retrieval. Effective software component retrieval from a repository is the aim of the ongoing research. In software component repository thousand of components are stored using various classification techniques. Software components have certain attributes associated with them, and each attribute has relative importance for that component, which is called as weights. Retrieving the components considering these features becomes more difficult and time consuming. The focus is to select the best component from a repository that can be reused. In the proposed system a component repository is developed and best components are retrieved in a two step process. The first step gives all the relevant components, and the second step gives the best component to the user. The first technique used is simple keyword based retrieval and second technique is genetic algorithm. Genetic algorithms give satisfactory results for those components which have attributes and weights. The genetic algorithms based technique is very effective when the repository size is very large.

This paper presents classification and retrieval of software reusable components by using a classification and also the selection of best component. The selection of best component or optimal section from retrieved components is much more difficult than simple retrieval. Finally experiments demonstrate results for displaying most relevant software reusable components.

Keywords: Software Reuse, Reuse Repository, Ranking algorithm, Integrated Classification.

INTRODUCTION

Software engineering is a profession dedicated to the creation and maintenance of software applications by applying technologies and practices from computer science. Programmers have always reused sections of code, templates, functions, and procedures. Software reuse as a recognized area of study in software engineering, however, dates only from 1968 when Douglas of Bell Laboratories proposed basing the software industry on reusable components. Software reuse is the use of engineering knowledge or artefacts from existing software components to build a new system. It is a technique which attempts to save time, energy and money by reducing the duplication of work. There are many word products that can be reused, for example source code, designs, specifications, architectures and documentation. The most common reuse product is source code.

Software Reuse is the process of implementing or updating software system using existing software assets. Those assets can be specifications, design, code, user documentation, or anything associated with software. Reusability is an important characteristic of a high quality software component [1]. Programmers should design and implement software components in such a way that many different programs can reuse them.

A software reuse library or component repository organizes stores and manages reusable components. With the fast growth of components repository, it is becoming difficult to retrieve relevant components. The biggest problem of software reusability in many organizations is the ability to locate and retrieve the existing software components. To overcome this problem, software components must be classified effectively for easier retrieval of components. The success of the software reuse programs depends on the classification technique used in creation of software reuse repository. This supports the engineers and other users in the process of developing the new software.

Five different classification techniques have been previously employed to construct reuse repository, they are Free-text classification, Enumerated classification, Attribute value classification and faceted classification scheme. The major drawback of free text classification is the ambiguous nature of keywords and search may result many irrelevant components. Enumerated classification is one dimensional and not flexible. This is not suited to classify software components. Attribute classification is multidimensional and no ordering of attributes. Faceted classification is easily expandable but making facets is difficult. There is no effective description and ordering of component in all existing classification schemes.

This paper focuses on development of a software tool which is to be used to classify and retrieve the software components as well as arrange all relevant software components with the help of a proposed ranking algorithm. Further the algorithm facilitates the user to select his desired component based on the calculated rank of a software component.

Section 2 reviews on related research work on the topic under investigation. The proposed software tool for classification and retrieval of software reusable components with ranking to select most desirable one is presented in section 3. The experiments carried out on the results obtained in the experiments and analyses them. The software tool presented in this work is concluded in this section 6 followed by references.

RELATED WORK

Existing Classification Schemes: A wide range of solutions to the software reuse classification and retrieval have been proposed and implemented. Software reuse classification and retrieval approaches
have been classified with minor differences based on available software reuse systems and also based on researcher’s criteria. Various methods to classify software reusable components have been proposed and implemented.

**Free text classification:** Free text retrieval performs searches using the text contained within documents. The retrieval system is typically based upon a keyword search [4]. All of the document indexes are searched to try to find an appropriate entry for the required keyword. An obvious flaw with this method is the ambiguous nature of the keywords used. Another disadvantage is that a search result in many irrelevant components. A typical example of free text retrieval is grep utility used by the UNIX manual system. This type of classification generates large overheads in the time taken to index the material, and the time taken to make a query. All the relevant text (usually file headers) in each of the documents relating to the components are index, which must then be searched from beginning to end when a query is made. Once approach to reducing the size of indexed data is to use a signature matching technique, however space reduced is 10-15% only.

**Enumerated Classification:** Enumerated classification uses a set of mutually exclusive classes, which are all within a hierarchy of a single dimension [5]. A prime illustration of this is the Dewey Decimal system used to classify books in a library. Each subject area, e.g. Biology, Chemistry etc, has its own classifying code. As a sub code of this is a specialist subject area within the main subject. These codes can again be sub coded by author. This classification method has advantages and disadvantages pivoted around the concepts of a unique classification for each item. The classification scheme will allow a user to find more than one item that is classified within the same section/subsection assuming that if more than one exists. This type of classification schemes is one dimensional, and will not allow flexible classification of components into more than one place. As such, enumerated classification by itself does not provide a good classification scheme for reusable software components.

**Faceted Classification:** Faceted Classification schemes are attracting the most attention within the software reuse community. Like the attribute classification method, various facets classify components, however, there are usually a lot fewer facets than there are potential attributes (at most, 7). Ruben Prieto-DiaZ [6, 10] has proposed a faceted scheme that uses six facets.

1. The functional facets are: Function, Objects and Medium
2. The environment facets are: System type, Function area, Setting
3. Enumerated classification: Fastest method, difficult to expand.
5. Free text classification: Ambiguous, indexing costs.
6. Attribute value classification: Slowest method, no ordering, and number of attributes.

**PROPOSED MODEL**

Information retrieval from components repository is a tedious work. The size of repository is often very large. Repository contains large number of components and its specification.

Repository is a link between developments for reuse, where the components are produced, and developed with reuse, where components are reused. For effectively reusing the components from the repository, the selection of proper retrieval technique is essential. Various linear searches based for information retrieval. In the proposed system we are using attribute classification scheme for classification and retrieval of software components. Every component is stored in repository with defined attributes. In the proposed software tool first step is to classify the software components employing a classification scheme based on attributes. In the next step all the relevant components are ranked according to a proposed ranking algorithm. This section presents a proposal for development of a software tool to classify and retrieve the software components as well as arrange all relevant components as well as it allows to select most desirable software reusable component with the help of ranking algorithm.

**User Request**

**CLASSIFICATION AND RETRIEVAL SCHEME**

The method of classification and retrieval of software components from the reuse repository is divided into two steps. The first step is the component classification phase, which sub divided into the encoding and classifier discovery phases. The second step explains about the retrieval of components from the repository.
Reusable Component Classification Method:

In the component classification phase the components are classified using the 9 characteristics. The classification scheme uses a genetic algorithm which evolves the small number of classifiers by dividing the set of available components stored in the reuse repositories into certain subsets. The innovative way of classification of components using the genetic algorithm will result in the fast retrieval of the correct components according to the requirements that are specified by the user.

This phase is further divided into two sub phases, the encoding phase and the classifier discovery phase. In the encoding phase components characteristics are encoded to binary form that is understood by the genetic algorithm. The genetic algorithm processes the bit strings and evolve the classifiers where each classifier holds the set of homogeneous software components.

Component Retrieval Method

In the component retrieval method, user will search for a specific component. First the user will enter the desired characteristics of components which he wants to retrieve from the reuse repository, through an interface. Second the user will set the matching threshold value (obviously the lower the threshold value the more components will be returned and higher the threshold value, exactly the components that matches user entered characteristics will get returned).

The system will encode the user request as a bit string and will compare it against all classifiers that were discovered in the classifier discovery phase. The closest match will signify the “winning” classifier and the components that are classified under the winning classifier will get returned.

GENETIC ALGORITHM BASED RETRIEVAL

A dedicated Genetic Algorithm [2] was developed to evolve candidate classifiers and select the optional solution in terms of number of components in the corresponding classes, which works in discrete steps as follows:

1. Create a random population of 100 chromosomes (potential classifiers)
2. For every generation of genetic algorithm
   2.1 Apply crossover operation to every pair of classifiers, where each pair is randomly selected according to the crossover probability
   2.2 Apply mutation to a randomly selected classifier according to mutation probability
3. Perform component classification for each of the 100 classifiers:
   a. Compare each classifier’s values of characteristics with those of each component. If component is close enough (determined by a threshold) to classifiers then assign the component to the class represented by this classifier.
   b. Select the top 20 classifiers (chromosomes) in terms of the number of assigned components. Then find the average numbers of assigned components to 20 classifiers. This is the average fitness of current generation.
   c. If the average fitness of the current generation is greater than that of the previous generation then create a new population by selecting chromosomes according to their fitness and repeat step 3. Otherwise do not create new population and repeat the step 2.

The above algorithm is repeated until a termination condition is reached. In our case the algorithm terminates if no improvement is the average fitness of the population is observed for 100 generations. A very important parameter is the value of threshold, which determines whether a component belongs to a certain classifier. For example, a value of 40% means that at least 40% of the values of the classifier characteristics are identical to those of a component. This threshold value essentially determines the “success” level of a classifier to gather a rich number of components in this class.

POPULATING THE DATABASE

Here user first uploads the components into the Repository. Each component has Attributes like name of component, operating system, time complexity, language implemented, inputs, outputs, domain, space complexity, version etc. Populating the database means storing the component in the repository. Administrator should be able to upload his components into the repository. While uploading the components the system should make sure that components are not duplicated. If there are two components with similar properties they should differ at least in the language, functionality or version and so on. If any component is left empty it is then filled with default values. The default values are already defined by the administrator.

CLASSIFICATION OF THE COMPONENTS

The classification scheme, which employs a combination of one or more classification techniques, is proposed and likely to enhance the classification efficiency. The proposal is described in the following sub section. This had given rise to development of a software tool to classify a software component and build reuse repository. Integrated classification scheme which combines the attribute value and faceted classification schemes to classify components with the following attributes.

RETRIEVAL OF THE COMPONENTS
This part of the model provides an interface for the user to search for a particular software component. The user can continue his search by providing a maximum of 3 attributes pertaining to a component. This information is once again processed on reuse repository. Here the Relevant software components from the reuse repository are retrieved by attribute value classification method. Relevant components are obtained by a query that is generated.

Components indexation and selection can be seen as information retrieval problems. Let us consider a component as a document on which we can apply all information (documents) retrieval techniques. The first works tried to apply plain text classification techniques to components retrieval problems (Boolean model [WAL79, SAL 83a], Vectorial model [LHU57, BUC92, ROB94, SIN96], Probabilistic model [MAR60, BOO74, ROB76, CRO79, TUR91], Linguistic model [SAL83, DEF86, NIE90a, NIE90b]). These techniques can be useful with some kind of components like patterns [ALE77, GAM95, and COP95] but they are not really helpful for applications engineer to select software components. For example the plain text classification techniques are not able to resolve queries such as “select all the components using Java technology which implement the Interface-name interface”. Such queries require component code analysis and externals information’s given by humans. Some components retrieval approaches were introduced for these specific needs. These approaches can be classified into three categories based upon the way components are represented: external classification, structural matching and behavioral retrieval.

APPLYING GENETIC ALGORITHM TO THE OBTAINED RELEVANT COMPONENTS
After the retrieval of relevant components these components are passed as input for genetic algorithm. Genetic Algorithm functions on fitness function value which is calculated using the following formula.

\[ \sum_{i=1}^{10} Ga_i V_i \]

Vi denote weight vector corresponding to that attribute vector, where i denote total number of attribute in each chromosomes, i.e. 10. The retrieved components undergo certain operations like crossover, mutation and selection.

OBTAINING THE BEST FIT COMPONENT
To select best fit component among all retrieved relevant components only three attributes are chosen for the application of genetic algorithm. All the retrieved components chosen attributes are assigned weights as per the pre-defined weight age. An encoding of components is required to process genetic algorithms. After the number of iterations by Genetic Algorithm on the relevant components we get a single component as output which is to be called as BEST FIT component for the attributes given by the user.

CONCLUSION AND FUTURE SCOPE:
Effective retrieval of software component from a software reuse repository is difficult and time consuming. In the absence of proper retrieval mechanisms the software reuse becomes ineffective. The main objective of this research work is to classify and retrieve the software reusable components from software reuse repository most efficiently. Further we had proposed and realized a ranking algorithm. The new classification scheme is realized in this work with utmost precision and accuracy for most efficient classification and retrieval of best suitable software reusable components.

Genetic Algorithms are best suited for large search space; proposed technique has been tested with 5000 components, which can grow to any size. Keyword based retrieval can be combined with another information retrieval technique to improve retrieval performance for candidate components. Genetic algorithms can be combined with other soft computing technique to get more optimized result. Text description of the components can be enhanced, for better results in the future.

REFERENCES
Jeffrey S. Poulin and Kathryn P.Yglesias, Experiences with a faceted Classification Scheme in a Large Reusable Software Library (RLS), In The Seventh Annual International Computer Software and Applications Conference (COMPSAC’93), Phoenix, Pp. 90-99, 3-5 November (1993)