Factors Affecting Retrieval of best Component from Repository

[1]Nisha, [2]Karambir
[1]Dept. of Computer Engineering, University Institute of Engineering and Technology, Kurukshetra University, Kurukshetra, Haryana, India
[2]Dept. of CSE, University Institute of Engineering and Technology, Kurukshetra University, Kurukshetra, Harayana, India

Abstract
Component Based Software Engineering (CBSE) is a process that emphasizes the use of reusable software components. There are many benefits of reuse like risk reduction, productivity increase, quality and reliability increase and reduction in cost and development time. Software reuse based upon component is now become basis of almost every software product line. There are many methods for representation and retrieval of components, but no one provide uniform formulism for component retrieval. In the paper we are presenting what are the main factors/issues for effective retrieval of appropriate component from repository.

Keywords
Component Reuse, Component Classification, Searching, Facet, Reuse and Optimization

I. Introduction

Software reuse is the use of engineering knowledge or artifacts from existing software components to build a new system [1]. There are many work products that can be reused, for example source code, designs, specifications, architectures and documentation. The most common reuse product is source code. A fundamental problem in software reuse is organizing collections of reusable components for effective search and retrieval. To retrieve the component from repository efficiently and effectively many factors should be consider. Before discussing these factors let us take look on what are the issues of software reusability which are shown in figure 1 and description of CBSE process is shown in fig. 2.

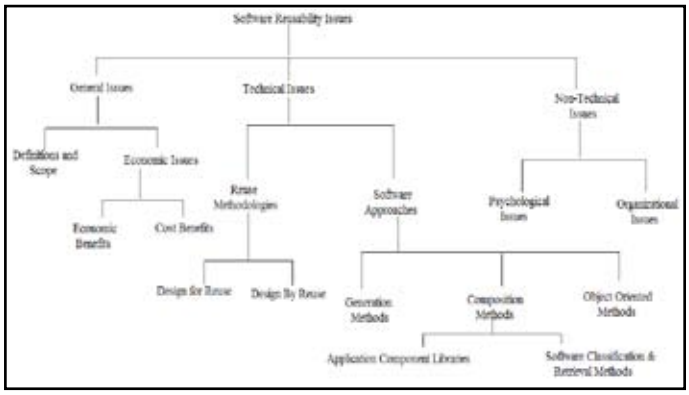

Fig. 1: Software Reusability Issues

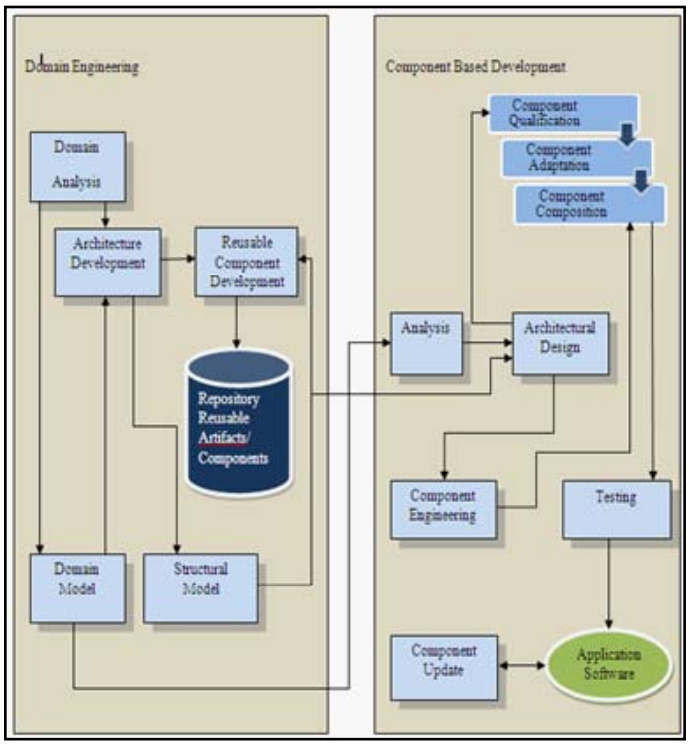

Fig. 2: CBSE Process

II. Query Formulation

A software component retrieval system can be considered as a tool for finding reusable software components. Like we have many search engines for formulation of our query, but developers who are not familiar with these types of search engines, often found troubles to retrieve reusable components with search engines. One of reason for this problem is construction of "good" queries. There should be refinement of search queries based on try and error, so that finding components suitable for one's requirement from large repository is become easier process [2].
So the first and foremost step for effective retrieval of software components is that query should be good and refined.

III. Structure of Software Repository

Software repositories play vital role in software reuse. A software library is a set of software assets that are maintained by an organisation for possible browsing and retrieval [3]. A component repository system should be independent enough so that information can be retrieved abstractly thought of as secondary information storage from the perspective of computer users, because information stored in this is accessible only after users have stopped working on their current tasks and switched from their workspaces. The retrieval process finds the components that match given reuse queries. Its desirable for an effective retrieval of components, a good representation schema for indexing and a matching criterion between a query and a component is essential. The "intelligence" of matching algorithm is of no use if components are indexed or

structured poorly [4]. So the structure of a repository is generally regarded as key to obtaining good retrieval results.

IV. Representation Schema for Indexing/Storage

Component Representation models have potential to make easier component search and selection. As such models can adopt high level concepts to describe components, making possible to semantically express their features [5]. In such context, XML-Based Asset Representation Model [6] is adopted to describe not only the components but also various assets produced on a component based development process. So to retrieve the components efficiently the components should be represented or indexed properly, so that they can be discovered and reused according to their features.

V. Classification Scheme for Components

Any reusable software item is generally known as a component. Components may consist of, ideas, designs, source code, libraries and testing strategies. But before retrieving a component of user's interest, the developer has to specify what components or type of components they want. So there is need of proper classification scheme for component. There are many techniques for component classifications. Existing techniques:

A. Free Text Classification

This is the most classical approach to retrieve the component i.e. to classify items by keywords. This retrieval system is typically based upon a keyword search [7]. All of the document indexes are searched to try to find an appropriate entry for the required keyword. An example of free text retrieval is the grep command of UNIX operating system. This type of classification makes large expenses Keyword approaches tend to miss potentially useful components because the people who classify the component in the library can not anticipate all potential of each component [8].

B. Enumerated Classification

This classification scheme uses a set of mutually exclusive classes and thus forms a single dimensional classification [9]. An example of this is the Dewey Decimal system that is used mostly for classification of books in a library [10]. Each subject area, e.g., physics, computer, chemistry related etc, has its own classifying code. sub code further represents the a specialist subject area within the main subject. These codes can further sub coded by year of publication etc.

C. Attribute Value

This scheme uses set of different attributes to classify a component. For an example, a book has many attributes such as the author name, publisher, year of publication, a unique ISBN number and classification code in the Dewey Decimal system. This entire can forms possible attributes for classifying a unique book [11].

D. Faceted

Prieto-Diaz has proposed another classification scheme for component using facets, which are groups of related terms in a subject area [12]. This approach better structures the terms used for classifying the components. Terms chosen from a set of facets are used to categorize all the components. This facilitates a closer fit of terms and reduces the problem of deciding the best keyword to use from a fixed set of standard keywords [8].

E. Browsing

Another classical approach for component's classification is browsing. Browsing systems depend on links among the items to be searched, and upon the user following those links to find the desired item. Experience shows that browsing through large structures can be very frustrating and time consuming [8].
We can also classify components by various another classification schemes like by formal Specifications, through knowledge- Based Approach, Signature Matching.

The advantage disadvantage of all these schemes can be summarised as:

Table 1:

Classification Scheme	Advantage	Disadvantage
Free Text Classification	Fast and easy to implement, Users can submit any query	Ambiguous, indexing costs, Generally produces results with low precision and low recall.
Enumerative Scheme	Fastest method	difficult to expand
Attribute Classification	no ordering	Slowest method
Faceted Classification	Easily expandable, Users can freely choose features available for search	Components must fit the classification scheme otherwise irrelevant components are retrieved, Some components may overlap categories
Formal Specifications	Automated, more precisely characterize functionality	Difficult to understand
Knowledge- Based Approach	More powerful than traditional methods	Require enormous human resources, manually populated
Signature Matching	Provides good information about program functions, good for searching components based on source codes	Only suitable for strongly-typed programming languages, not good for metadata search

VI. Component Searching Mechanisms

After entertaining a refined query from the user, proper maintenance of repository done and effective classification scheme, the next factor that influence the effectiveness of component retrieval is what kind of searching mechanism we are using to locate the component. It will be totally frustrating experience when time to search the component exceeds. So there should be proper searching mechanism. Reusable software should be retrievable. For this we should arrange the libraries of reusable modules and provide a good database searching mechanism such that client programmers can find appropriate modules easily [13].

On the other hand reuse of software components in an appropriate manner is often a difficult task, particularly when the search and selection of components must be conducted over a large collection of components that, in general, are documented in an unsatisfactory manner. Further according to Frakes [3], to make reuse process effective, developers should be able to find components easily. Therefore, to search and select components suitable for developer's needs it is important there should be proper development of techniques and tools for information retrieval. As a key outcome, several search systems have been proposed in industry and academy to assist identification and selection of software components. Some of them adopt approaches in which components features are automatically extracted [4].

VII. Evaluation Criteria

After the selection of proper methods for component classification and searching, next step is to assess and compare storage and retrieval methods. There are many criteria for evaluation of these used methods like: to make evaluation on the basis of technical Criteria i.e. precision, recall, logical and time complexity of method, managerial criteria i.e. operating cost, investment cost, human criteria which include difficulty of use and transparency of method [16]. But to check whether component repository works effectively we usually evaluate the correctness and effectiveness so the recall and precision measures. Precision and recall are two concepts that have traditionally been used to evaluate the method of retrieval of software components.

Recall means to get all the relevant components

$$\text{Recall} = \frac{\text{No. of relevant retrieved assets}}{\text{Total No. of relevant assets in the library}}$$

Precision means that all the retrieved components are exact as per query submitted by a user.

$$\text{Precision} = \frac{\text{No. of relevant retrieved assets}}{\text{Total No. of retreived assets}}$$

Both of this value should lie in ranges between 0 and 1.Under the hypothesis that the entire library assets are visited, we get perfect precision (=1) whenever the matching condition logically implies the relevance criterion. But it is very difficult to get both high precision and recall using the classical approach. Also there is no criterion for evaluating the relevancy of components. For effective use, the keywords have to be independent, they have to span the range of the users need, and the users must be able to pick the right keywords [16].

VIII. Optimisation Technique

After considering all the above discussed factor for effective component retrieval, we can also optimised the search result by using various optimisation technique. The Ant Colony Optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The same can be used to find out exact component from repository [17]. We can also use various string matching algorithms for exact matching of component and can applied genetic algorithm for optimised searching results.

IX. Summaries

These are the primary factors that one should take care in order to achieve best component from repository. All these factors are coupled with each other tightly i.e. one can't retrieve the exact component if the query is not refined and formulate properly, if component are not indexed and represented properly, if searching mechanism used result in high time complexity. There is number of approaches that are currently available for retrieving appropriate component, but these approaches suffers from various shortcomings which result that there should be the need to use various queries like low level, service-based queries, lack of high-level description of component capabilities, lack of validation or checking of retrieved component suitability, and lack of use of the context for which queries are being performed by the retrieval tool. Future work can be done with these classification schemes so that these will be used to refine the scheme for Multi-Tired or Multimedia presentation of components. Another important aim of the repository system is to build capabilities to support concept generation. Also various ways of optimizing system speed and efficiency must be explored in order to keep the repository as an effective design tool.

References

[1] William B.Frakes et al.,"Software Reuse Research: Status and Future", IEEE transactions on Software Engineering, Vol. 31 No. 7, July 2005

[2] Makoto Ichii et al.,"Software Component Recommend-ation Using Collaborative Filtering", IEEE, May 2009.

[3] Mili, H et al.,"Reuse Based Software Engineering", Wiley-Interscience Publication, USA, 2002.

[4] S. Henniger,"Supporting the construction and evolution of component repositories", Proceedings of the 18th International Conference on Software Engineering (ICSE'96), Berlin, Germany, 1996.

[5] Brito,T., Nobrega,H., Ribeiro,T., Elias,G.,"A Search Service for Software Components based on a Semi-Structured Data Representation Model", IEEE, 2009.

[6] G. Elias, et al.,"X-ARM: An Asset Representation Model for Component Repository Systems", 21st ACM Symposium on Applied Computing, 2006.

[7] Ruben Prieto-Diaz et al.,"Building and Managing Software Libraries", IEEE,1998, pp. 228-236.

[8] Luqi et al.,"Toward Automated Retrieval for a Software Component Repository".

[9] Vicente Ferreira de Lucena Jr., "Facet-Based Classification Scheme for Industrial Automation Software Components".

[10] E.Smith, A.Al-Yasiri, M. Merabti,"A multi tiered classification scheme for component retrieval" , Proc. 24th Euro microconf.,1998, pp. 882–889.

[11] P.Niranjan, C.V.Guru Rao,"A mock- up tool for software component reuse repository", International journal of software engineering and applications (IJSEA), Vol. 1, No. 2, April 2010.

[12] Rubin Prieto-Diaz,"Implementing Faceted Classification for Software Reuse", Communication of the ACM, pp. 89- 97, May 1991.
[13] Patrick Shicheng Chen et al.,"On the Retrieval of Reusable Software Components", IEEE ,1993.
[14] W. Frakes, T. Pole,"An Empirical Study of Representation Methods for Reusable Software Components", IEEE Transactions on Software Engineering, Vol. 20, Issue 8, August 1994, pp. 617-630.
[15] M.R. Girardi, B. Ibrahim,"Automatic Indexing of Software Artifacts", 3rd International Conference on Software Reuse, 1994.
[16] Vaneet Kaur, Shivani Goel.,"Facets of Software Component Repository", IJCSE, Vol. 3, No. 6, June 2011.
[17] Sandeep G. Khode, Rajesh Bhatia,"Improving Retrieval Effectiveness using Ant Colony Optimization", IEEE, 2009.

Nisha was born in Haryana, India in 1989. She received her B.Tech degree in Computer Engineering from DIET, Karnal and M.Tech from UIET, Kurukshetra University and currently she is working as Assistant professor in NIT, Kurukshetra, Haryana.

Mr. Karambir is currently working as a assistant professor in Department of Computer Science and Engineering,University Institute of Engineering and Technology (UIET) Kurukshetra University, Kurukshetra. He did his B.Tech from Nagpur University and received his M.Tech degree from GJU, Hisar, Haryana.