AN EFFECTIVE RETRIVAL SCHEME FOR SOFTWARE COMPONENT REUSE

Vishal
Department of Information Technology
BPS Women University
Khanpur Kalan(Sonipat) India
vishal_pasricha@yahoo.Com

Subhash Chander
Research Scholar
Dravidian University
Kuppam India

Jasmer Kundu
BPS Women University
Khanpur Kalan(Sonipat) India
Jasmer_kundu@rediffmail.com

***Abstract*-Software component reuse has become of much interest in the software community due to its potential benefits, cost benefit, time saving, etc. which include increased product quality and decreased product development cost and estimated schedule. To select a component for reuse is very difficult because there are many components with approximate same name, same functionality etc. which make the reuse of component very time consuming. Hence there is required a technique which make the selection of component efficient and fast. For this purpose we purposed a scheme for the selection of component which will be more efficient than earlier scheme.**

I. Introduction

A. What is Software Reuse?

Software reuse is a process of creating software systems from the existing software rather than building them from low level design [1]. Software reuse is a fastest growing and very needy displine todays. Software reuse appears in various different forms from white-box reuse to black-box reuse, and from ad-hoc reuse to systematic reuse. in many different types software is reuse like SRS document, algorithm, design, source code, test case that are made during the software life cycle. Source code and tested modules are most commonly used. Many developer software reuse as the source code alone. Design reuse is popular in object oriented class libraries. The software community does not finalize what a software component exactly

B. *Benefits of Software Reuse*

Software reuse has a positive impact on software quality, costs, and productivity.

1. Quality Improvements

Software reuse results in improvements in quality, productivity, performance, reliability and interoperability.

a. Quality:- Error fixes accumulate from reuse to reuse. This produces higher quality for a reused component than would be the case for a component that is developed and used only once.

b. Productivity:- A productivity gain is achieved due to less code that has to be developed. This results in less testing efforts and also saves overall yielding's cost. When reuse is being installed, productivity may decrease shortly due to increased learning effort and the need to develop reusable components. This temporary decrease in productivity should easily be compensated by a long-term increase.

c. Performance:- This may yield better performance of a reused component than might be practical for a component that is developed and used only once. However, generalizations that make components more reusable can have a negative influence on overall performance.

d. Reliability:- Using well-tested components increases the reliability of a software system. The use of a component in several systems increases the chance of errors to be detected and strengthens condense in that component.

e. Interoperability:- Many systems can work better together if their interfaces are implemented consistently. They use the same components for these interfaces. Even though written standards improve interoperability, different implementations might differently interpret parts of these standards.

The quality of reused components (numbers of errors per lines of code) was about 9 times better during component test and about 4.5 times better during system test. In another project there were even no errors found in reused components during the entire life cycle of the project.

2. Effort Reduction

Software reuse reduces the work and development time, which yields to a shorter time to market. This is important considering the importance of good timing, i.e., early availability for software systems. Additionally, documentation costs as well as team sizes can be reduced.

a. Redundant Work, Development Time:- Developing every system from scratch means redundant development of many parts like basic algorithms, user interfaces, communication, etc. This can be reduced when these parts are available as reusable components and can be shared, resulting in less development and less associated time and costs.

b. Time to Market:- The success or failure of a software system is often determined by its time to market. By using reusable components we can get result in a reduction of that time.

c. Documentation:- The documentation is very important for the maintenance of a system, it is often neglected. Reusing software components reduces the amount of documentation to be written but compounds the importance of what is written. There is need to structure of the software system and the newly developed components have to be documented.

d. Maintenance Costs:- Fewer defects can be expected to occur when proven components have been used, and less of the software system must be maintained. The reusable components are maintained by a separate group rather than separately in each software system.

e. Team size:- Large development teams suffer from a communication overload. It is clear that doubling the size of a development team does not result in doubled productivity. If many components can be reused, then software systems can be developed with smaller teams, leading to better communication and increased productivity.

C. Why Reuse Software?

A software reuse process provides the facility the increase of quality, productivity and reliability, and the decrease in implementation time and development cost also. There is only investment in to start a software reuse process, but after that investment recovers in only few reuses. In short, the reuse process's development and reuse repository produces a knowledge that improves in quality in every reuse, minimize the amount of development cost, development work needed for future projects and the reduce in risk of future projects. A software component is an independent part of the software system having complete functionalities.

Four levels of component reuse have been proposed:

1. Design level products
2. Analysis level products
3. Code level components (modules, procedures, subroutines, libraries, etc.)
4. Entire applications

D. Types of Reuse

Components are used the following types is given below.

1. *Horizontal reuse*

Horizontal reuse refers to those software components that are used across a wide variety of applications in terms of code. Reuse of third-party application or modules within a system, such as an e-mail, Facebook access program. Various software libraries and repositories containing these code and documentation exist on the internet today.

2. *Vertical reuse*

The basic idea is the reuse of system functional areas, or domains that are potentially very useful, can be used by the systems with similar functionality [2]. Domain engineering is "achieve iterative, comprehensive, life-cycle process that an organization can use to strategic business objectives. It increases the throughput of application engineering projects through the normalization of a product family and a related production process [3]". Domain engineering concentrations on the formation and preservation of reuse repositories of functional areas.

3. *Systematic software reuse*

Systematic software reuse and the reuse of components impact almost the whole software development process [2]. Software process models were established to provide guidance in the creation of high-quality product at predictable costs. Software process models are adapted based on experience, changes and improvements were

suggested by classic waterfall model. New models are also based on efficient reuse of useful components that have been developed in other projects [2].

E. Software Component

1. Describing a Software Component

Component is fundamental unit of a software construction. Every component has his own interface and an Implementation. The following are the exclusive aspects of a Software Component that must not only be described, but also searchable:

- Version data
- OS/Platform compatibility
- Development
- Scalability
- Testing specification, and performance data
- Known deficiencies
- Functional specification
- Interface specifications
- Use cases, use scenarios
- Supportability data
- Evaluations

F. Reusable Software Component

A component is an autonomous part of software that interacts with other components in a definite manner to accomplish a specific task. Components can be used to build both shrink-wrap software and enterprise critical software [5]. Reusable software points to those software components can be combined with a variety of programs without modification. Reusable software components are designed to take benefit of reusability in software construction. Software reuse is the use of knowledge or artifacts from existing software components to build a new software system. There are numerous work products that can be reused, for example designs, specifications, architectures, source code and documentation [4, 6].

G. Existing Methods of Components Classifications

What is Component classification: reusable software item is generally known as a component. Components may consist of, ideas, designs, source code, linkable libraries and testing strategies but they are not necessary. In classification, the developer has to specify what components or what type of components they want. These components then should be retrieved from a library, evaluated as to their suitability, and improved if required. When the developer is satisfied that they have retrieved an appropriate component, then it can be added to the current system under development. The aim of a component retrieval system [5] is to make able to located either the exact component necessary, or the closest match, in the minimum amount of time, by using a suitable query. The retrieved component(s) should available for possible selection.

Classification is the process of selecting of interest. The classification of components for reuse is more difficult than classifying books in a library. Library systems always use structured data for its classification system (e.g., the Dewey Decimal number). Current classification process to classify software components divided into the following categories: Free text, enumerated, attribute-value, and faceted. The selection of each of the methods is evaluated as to how well they perform against the described criteria for a good retrieval system.

a. Existing techniques

1. Free text classification: Free text retrieval is based upon a keyword search. In this type classification technique, a user inputs keywords to search. The retrieval system does searches using the text contained within documents [7]. Indexes are searched to try to find a suitable entry for the required keyword and as a result a ranked list of documents is returned. The ambiguous nature of the keywords is major drawback of this method. Another disadvantage is that a search returns many irrelevant components as a result. An example of free text retrieval is the grep command of UNIX operating system. This type of classification makes large expenses in the time taken to index the components, and the time taken to make a query. All the relevant text in each of the documents related to the components are index, which must be searched from starting to end when a query is fire. Relevant keywords are derived by their statistical and positional properties, thus resulting is called

automatic indexing. Two processes indexing and searching are involved in keyword search. Keyword search provides a freedom to users to freely submit their query to search engines;

2. Enumerated classification: Enumerated classification is a single dimensional classification that uses a set of mutually exclusive classes [9]. An example of this is the Dewey Decimal system that is used in a library to classify books [8]. Each subject area, e.g. mathematics, physics, computer 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 author or publisher. This classification method has their own advantages and disadvantages pivoted around the concepts of a unique classification for each item. The classification scheme allow a user to find more than one item that is classified within the same division/ subdivision assuming that if more than one exists. For example, there may be more than one book of a subject, each published by different publishers. Major problem in this type of classification schemes as is one dimensional do not allow flexible classification of components into more than one place for reusable software components. However it provides significant support for best effort retrieval of components.

3. Attribute value: The attribute value classification scheme uses set of attributes to classify a component [10]. For an example, a book has many attributes such as the publisher, the author name, a unique ISBN number and classification code in the Dewey Decimal system. These are the only example of the possible attributes. Based on the search query of reader a book can be combed in no. of ways e.g. number of pages, the size of the paper used, the type of print face, the publishing date, etc. clearly, the attributes relating to a book can be:

- Bulky. All possible combinations of attributes could run into many tens, which may not be known at the time of classification.
- Multidimensional. The book can be classified by different attributes in different places.

4. Faceted: Faceted classification was proposed by Prieto-Diaz and Freeman in 1987[11] also known as faceted navigation or faceted browsing that relies on facets which are mined by experts to describe the features about components ex:- Features, such as the component's functionality, how to run the component, and implementation details. Like attribute classification method, various facets are used to classify components however there are usually a lot fewer facets than there are potential attributes. Sometimes users are given choice to select features for search. This helps the users to achieve his search goals rapidly and efficiently. Faceted classification and retrieval has verified to be very effective in retrieving reuse component from repositories.
Ruben Prieto-Diaz has proposed a faceted scheme that uses six facets.

- The functional facets are: Function, Objects and Medium.
- The environmental facets are: System type, Functional area, setting [1].

Each facet has to have values assigned at the time the component is classified.
Faceted classification scheme is very attracting the most attention within the software reuse community. Like the attribute classification method, facets classify components however there are generally a lot fewer facets than there are potential attributes.
Each of the facets has values assigned at the time the component is classified.
The individual components can then be distinctively identified by a tuple,
for example. < add, arrays, buffer, database manager, billing, book store >
Frakes and Pole conducted an investigation on the above classification methods [9]. The investigation found no any major differences between the four different classification schemes; however, the following about each classification method was noted:

- Free text classification
  Ambiguous, indexing costs
- Enumerated classification
  Fastest method, difficult to expand
- Attribute value classification
  Slowest method, no ordering,
- Faceted classification
  Easily expandable, most flexible

Classification Schemes/Characteristics	Ambiguity	Flexibility	Speed	Indexing Cost	Expend Quality	Index Ordering
Free Text Classification	YES	Moderate	Average	High	Normal	YES
Enumerated	YES	Least	Fastest	More	Difficult	YES
Attributed Value	NO	Normal	Slowest	Less	Normal	NO
Facet Classification	NO	Most	Slow	Very less	Easily	NO

Table 1. shows the comparison among all the four existing component selection scheme.

II, PROPOSED SYSTEM

Various kinds of classification schemes are available to classify a component but one problem is how to implement this, many systems follow the principle: Once a new component has been identified, an administrator is responsible for the classification should be proficient. The system is employed for two reasons. First, the administrator must know how to classify the components according to the scheme. Second, a lexicographical consistency is necessary across the whole of the system. The classification system is always separate to the retrieval system, which is for the users. Well-known systems works with one classification and retrieval scheme, such as enumerated or faceted. Others work with a different classification system.

In this research, we propose a new classification scheme that integrates the features of existing classification schemes.

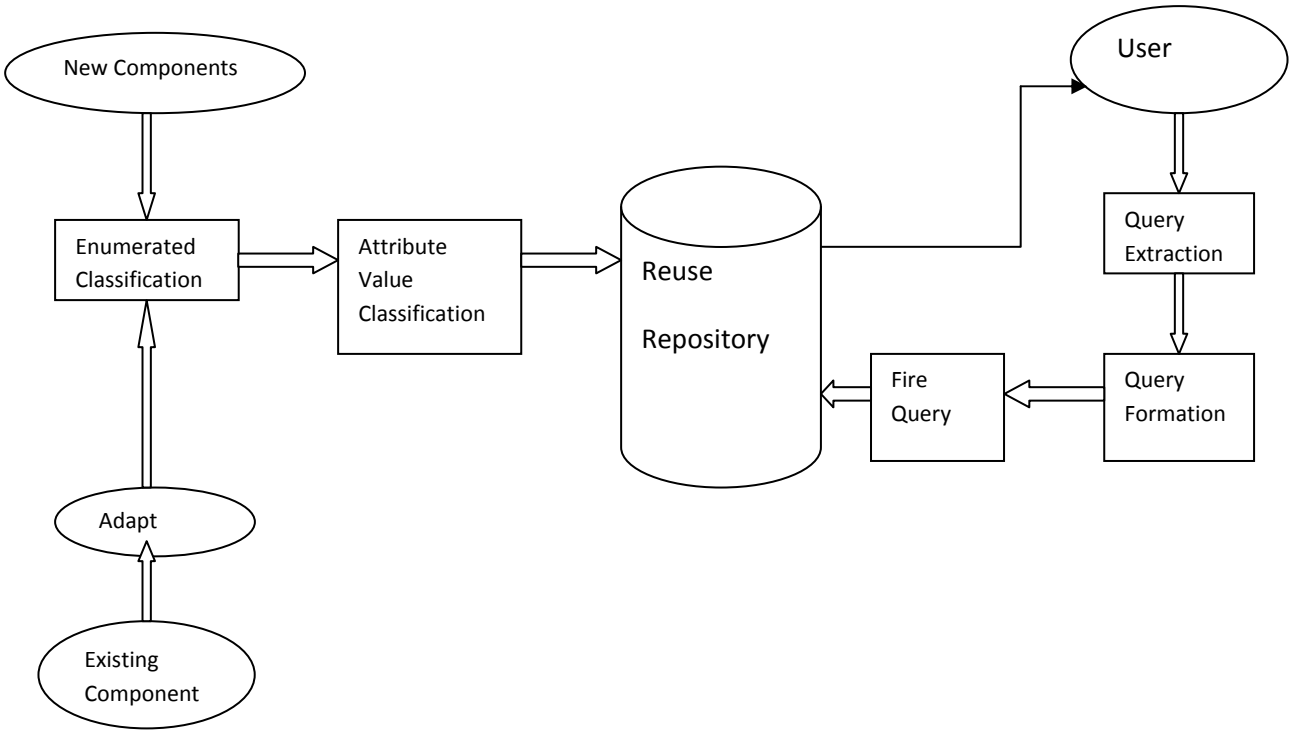


Fig 1. Shows the new efficient component retrieval scheme for component classification

In this system (Fig.1), we set up the proposed scheme. Here the developers develop a new component and put their components into library at the same time if required the developers can also retrieve components from the library. Query tracking system should be maintained to improve the proposed scheme. The proposed system will provide the following functionality to the users.

- Storing components:- To store the newly developed and maintained components in the reuse repository for future use.
- Searching components:- This functionality is given to user to search the available components from the libraries.
- Browsing components:- This functionality is used to extract the copy of the available component for the reuse and modification from the libraries.

A. Proposed algorithm

Step1:- User enter the query through the interface

Step2:- Build the query and apply the enumerated classification scheme using the query on the reuse repository.

Step3:- Store the results extracted by enumerated classification in temporary buffer.

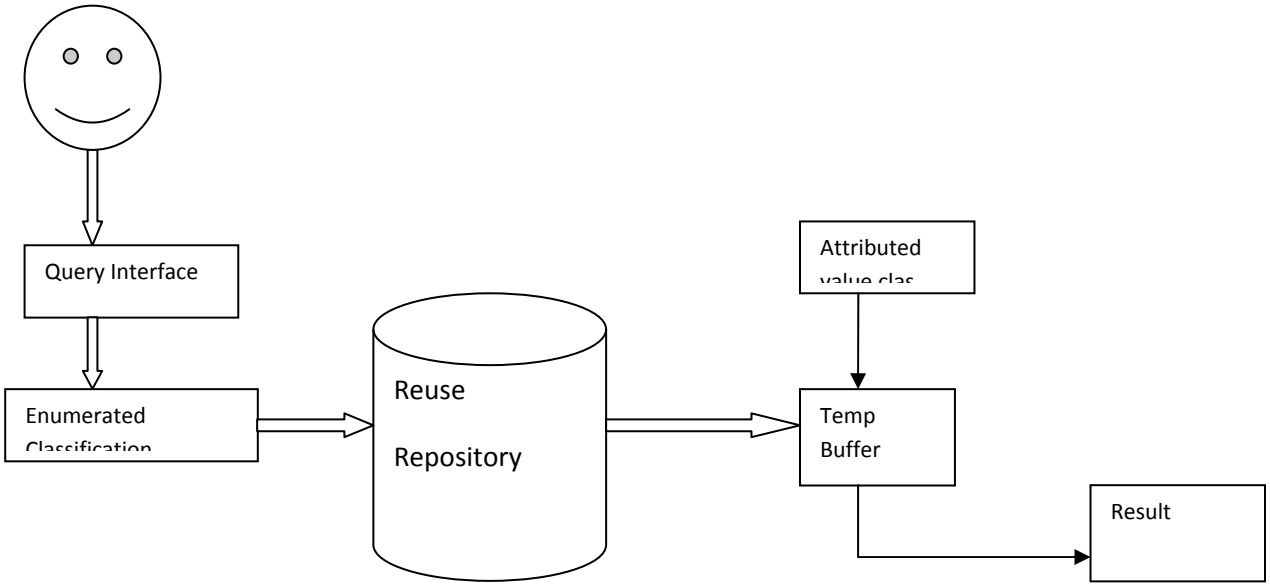
Step4:- Now apply the attributed value classification on the results available in buffer.

Step5:- The final result which comes after the attributed value classification is returned to user.

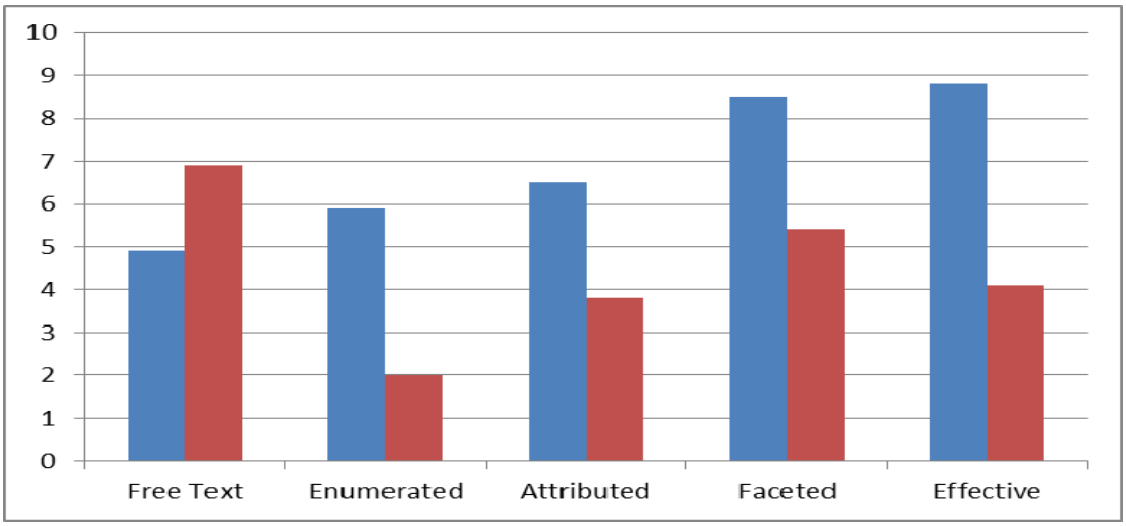
Each of the four main classification schemes has both advantages and disadvantages. The free text classification method does not offer the flexibility and has many problems with synonyms and search spaces. The faceted classification method offers the most flexible method of classifying components and has problems when trying to classify similar components for use. The enumerated classification scheme provides a fast way to drill down into a library, but does not offer the flexibility to classify components for use in more than one way. The attribute value classification scheme permits multidimensional classification of the same component, but does not offer any ordering of the different attributes.

The efficient classification scheme took the advantages or positive sides of both schemes i.e. enumerated classification and attributed values classification. Our solution to these problems would be to use an enumerated classification scheme combined with attribute value classification scheme to classify the components details. The enumerated scheme is initially used to reduce the search domains. By using the enumerated classification scheme we make the search space to be restricted to specific points. The components selected by the enumerated classification scheme are the available components in the buffer are considered for attribute value scheme. By reducing the size of the search domain attribute value have to search in limited domain so it gives the more accurate results and efficiently. The particular component is then classified using attribute value scheme.

Figure 2 shows the architecture of proposed system. The repository is library of components and their information. Users are provided interface through which they can interact with components. To upload a component User will provide the details. To retrieve a component the user will give his query or he will give details so that the system will provide the matching components.



iii. Experimental results



Graph 1. Shows efficiency to relevant components in blue color and their corresponding Search time in red color

We implement all the classification schemes's prototype in Matlab. We analyse their efficiency in searching relevant components and their search time to find relevant component. In the graph we can see that the new scheme known as robust scheme is providing most efficient results whereas its searching time is also high. But its provide the components more correctly which is demand of today's market. Hence by see result we can say that it is efficient technique for fulfilling the demand of market.

iv. Conclusion

In this paper four classifications (free text, attribute value, enumerated and faceted classification) are examined and opened their advantages and disadvantages. The effective system take advantage of two classification scheme. An effective software tool with user friendly interface is developed with integrated classification scheme which restricts search space and reduces search time increasing the efficiency of classification of software component and also showing the good results when its prototype is implemented on Matlab.

V. References

[1] Charles W. Krueger Software Reuse "ACM Computing Surveys (CSUR) Volume 24, Issue 2 (June 1992) Pages: 131 - 183.
[2] Sametinger, Software Engineering with Reusable Components, Springer-Verlag, ISBN 3-540-62695-6, 1997.
[3] Department of the Navy. DON Software Reuse Guide, NAVSO P-5234-2, 1995.
[4] B.Jalender, N.Gowtham, K.Praveen kumar, K.Murahari, K.sampath "Technical Impediments to Software Reuse" International Journal of Engineering Science and Technology (IJEST) , Vol. 2(11),p. 6136-6139.Nov 2010.
[5] M. Pat Schuler, "Increasing productivity through Total Reuse Management (TRM)," Proceedings of technology2001: The Second National Technology Transfer Conference and Exposition, Volume 2, Washington DC, December 1991, pp. 294-300.
[6] M. Aoyoma, "New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development?," in Proceedings of the 1998 International Workshop on CBSE,.
[7] Gerald Jones and Ruben Prieto-Diaz, "Building and Managing Software Libraries", © 1998 IEEE, pp.228-236.
[8] E. Smith, A.Al-Yasiri and M. Merabti, 'A multi tiered classification scheme for component retrieval'. Proc. 24th Euro micro conf.,1998, pp. 882–889.
[9] Vicente Ferreira de Lucena Jr., "Facet-Based Classification Scheme for Industrial Automation Software Components"
[10] Ruben Prieto-Diaz, "Implementing Faceted Classification for Software Reuse" © 1990 IEEE, pp.300-304
[11] Gerald Kotonya, Ian Sommerville and Steve Hall, "Towards A Classification Model for Component Based Software Engineering Research", Proceeding of the of the 29th EUROMICRO Conference © 2003 IEEE

Authors Profile

Vishal is currently working as a assistant professor in BPS women university, Khanpur Kalan(Sonepat). He did his B.Tech from Kurukshetra university, Kurukshetra with 69.92%. He did his M. Tech from YMCA university, Faridabad with 77.6%. His area of interest is Software Engineering, web crawling. He has also published papers in the field of web crawling.

Subhash chander is pursuing PhD(research scholar) in computer science from Dravidian university, Andhra Pradesh. He did his MCA from Kurukshetra university, Kurukshetra. He is also university medalist(K.U.). His area of interest is web crawling and software engineering. He has also published several papers in the web crawling.

Jasmer Kundu is currently working in BPS women university, Khanpur Kalan (Sonepat). He did his MCA from Rajasthan Univ. and M.Tech from Karnataka State open University. His area of interest is Artificial Intelligence and Software Engineering.