Component Retrieval Techniques-A Systematic Review

NIDHI
Nidhiruhil90@gmail.com
ITM University (Gurgaon)

AMAN JATAIN
Amanjatain@itmindia.edu
ITM University (Gurgaon)

Abstract: With the development of the component technology and the expansion of component library, representing and retrieving components has become a major issue to reuse the components. Many papers have been published providing various techniques but none of the paper has given a systematic review of these techniques. This study discusses the current state of the reusability techniques for the retrieval of components. The result of the review is classified on the type of approach and the type of approaches to validate the approaches.

Keywords: Software reusability, software component, review.

1. INTRODUCTION:

Software component reuse is the use of existing software components to build a new software system. Effective storage and retrieval of software components is much essential in software components reuse process. A software component is an independent part of the software system having complete functionalities. Every component has its own interface and an Implementation.

Reuse deals with the ability to combine separate independent software components to form a larger unit of software. To incorporate reusable components into systems, programmers must be able to find and understand them. The 3 parties are involved, one specifies the component, one implement the specification of a component and another one deploys or uses the component. Components can be used to build both shrink-wrap software and enterprise critical software. Reuse is the key paradigm for increasing productivity and quality in software development. To be able to reuse software components, whether it is code or designs, it is necessary to locate the component that can be reused. Locating components, or even realizing they exist, can be quite difficult in a large collection of components. It is desirable to have a repository that supports the storage, query and retrieval of software components and makes reuse possible. Most existing software component repositories only retrieve a limited set of Software components and some do not satisfy user queries. Finding and reusing appropriate software components is often very challenging particularly when faced with a large collection of components. By understanding the component reuse historical data and the feedback data of other re-users' component retrieval Component retrieval makes re-users understand the components better, and understand the focused components' information in practical application. Some of the major issues that make software reuse difficult are component classification, retrieval, and composition and library maintenance.

Research in these areas is needed to attain the potential increases in productivity, quality, and reliability. Literature [1] published in 1991 focuses on computer-aided retrieval of reusable software components. Literature [2, 3, 4, 5, 6, 7] includes Browsers, informal specification and formal specification to retrieve the components. Literature [8] published in 1995 gave attention on the case based retrieval of components**.** Case-Based approach enables reasoning about concepts, allowing the retrieval of "approximate" components. Literature [9, 10] focuses on the external classification of components to retrieve the components. External classification category includes all the approaches that represent the component by an external description. Literature [11, 12] published in 1997 specified an approach that encoded the descriptive information of reusable software components. A Literature published in 2001 presented a model- independent, multilevel, closed loop approach to software reuse that makes tuning of software reuse mostly automatic, and includes a reward system for development of reusable components. It is mostly suitable for the development in small. Literature [13] focuses on an optimization approach to plan for reusable software components. This approach presents a model that selects a set of software components that must be built, in order to lower development and adaptation costs. Then a literature presented full text indexing to retrieve the components from the repository. This gives comparable precision but much better recall than controlled vocabulary indexing of components. Literature [14] proposed a Conversational Component Retrieval Model (CCRM).

In CCRM, components are represented as cases, a knowledge-intensive case-based reasoning (CBR) method is adopted to explore context-based semantic similarities between users‟ query and stored components, and a conversational case-based reasoning (CCBR) technology is selected to acquire users‟ requirements interactively and incrementally. Literature [15] focuses on a novel approach as rough fuzzy approach. This approach selects the candidate components used in various previous projects for reuse in current applications. The methodology is based on rough–fuzzy sets, and a decision support tool based on this methodology is implemented. Literature [16] published in 2004 presented retrieval of components using genetic algorithm. It is a two step process. The initial step is Keyword based retrieval for finding all candidate components according to the user requirement. Keyword based search is used to narrow down the search space, which further improves initial population for Genetic Algorithm based second step. Feature space is adopted to uniformly describe semantics of components and business requirements; therefore the problem of component retrieval is transformed to the problem of feature tree matching. This method eliminates some deficiencies in traditional component retrieval methods. Literature [17] purposed a classification scheme as intelligent **classification scheme in 2006** to retrieve the components.

This method for the classification and retrieval of components from the reuse repository is divided into two phases. Literature [18] proposed in 2006 specification based approach was used to retrieve the components. Specification matching relationship has two levels: business operation level and business component level. Then a Literature suggested free text and faceted classification in 2007 for the retrieval of components. Literature [19] published in 2008 focuses on the structural and behavioral classification of components to retrieve the components. The basic notion of this approach is to combine formal and semiformal specification used to describe component and then to make the retrieval process more efficient.

2. RELATED WORKS:

The function of a software component retrieval storing system is that construct the model of software component retrieval, in the model, functions, apply domains, work environments, working static and dynamic behaviors of a software component can be accurately expressed. A software component includes the entity, describing and metadata information in a software component repository. The three can be stored together or discretely. There are two popular representation schemes for software components. The first is based on a controlled vocabulary, which is arranged in a strict hierarchical way. This scheme has received a lot of criticism as being inflexible and not producing a good classification. The second approach postulates a representation which is based on a series of characteristics (attributes) that assume values. This scheme is more flexible with regards to expansibility, which we have also adopted in the current work. As the component repositories scaling up and the reuse practice deepening, querying and retrieving components with high efficiency and accuracy to satisfy specific business requirement for short software delivery time and high software quality gains more attentions from software engineering researchers. There has been a large quantity of component retrieval algorithms in literatures, such as behavior sampling based retrieval, signature and specification matching based retrieval, etc.

3. Research Methodology:

Various researchers provide different techniques but none of the paper has given the systematic review of the paper.
In this paper we have followed an adapted form of systematic review process.

Question1: What approaches has been introduced to retrieve the components?
After specifying the question, a review protocol is used. The review protocol (Table 1) includes the sources used in the review, the period of the time under review.Table2 discusses the approaches to retrieve component.

IJSER

TABLE1**:** Review Protocol.

YEAR	SOURCES
1995-2013	Google scholar, IEEE Xplore

TABLE2**:** Approaches used to retrieve components.

	Approaches	**Papers**
Question 1: What approaches has been introduced to retrieve the components?	Browsers	1
	Informal Specification	1
	Formal Specification	1
	Case Based Retrieval	1
	External classification	1
	Encoding the components descriptive information	1
	Optimization approach	1
	Full text indexing	1
	Conversational case based reasoning	1
	Rough fuzzy approach	1
	Genetic algorithm	1
	Intelligent classification scheme	1
	Specification based approach	1
	Free text	1
	Behavioral and structural classification	2
	Attribute	2
	Total	18

3.1 Browsers technique:

A browser [1] is a general purpose, usually a window based tool for looking through collections, categories, hierarchies of components at various levels of abstraction,. The interface can range from purely textual to sophisticated graphics. In any case, the objective is to allow the system user to manually search for the desired component. The notion of a browser comes from the information retrieval domain, but its first use with respect to component retrieval was in object-oriented programming systems.

In an object-oriented system, reusability is inherent because all new objects are defined in terms of other objects already defined in an object hierarchy. It would be nearly impossible to manage this type of programming environment without some method to scan the hierarchy of components to find a suitable "jumping off point". Thus we see sophisticated graphical browsers for object-oriented systems like Smalltalk-80, the Knowledge Engineering Environment (KEE), and Eiffel.

The advantages of a browser are that it gives the user free reign over the entire collection of components, and in object-oriented programming systems allows the user to see which objects depend on other objects. There are, however, several disadvantages to the browser approach. The first is that the method is basically manual, relying on significant user knowledge of the structure of the component collection. Second, the focus of search is local, meaning that a semantically similar component defined elsewhere in the system will not be found at all unless the user knows to look there also. Third (and this is related to the second point), unless the user has found exactly the component needed, they will not know when to stop looking.

3.2 Informal Specification technique:

Retrieval techniques based on informal specifications require the user to describe or list some of the attributes of the component sought. Informal specification methods include keyword search, multi-attribute search, and natural language interfaces.

- **Keyword Search:**

Keyword search mechanisms [2] require the user to specify a list of words relevant to the object being sought. For example, if a user were searching for a component that implemented a stack, use the keyword *stack* to perform the search. One problem with using keywords is that the number and choice of words is crucial to success. Using a single keyword will often result in high recall but low precision, whereas too many keywords will have the opposite effect. The search for a component, then, becomes an exercise in trial and error, with the user performing multiple searches until an appropriate object is found. It often takes an experienced user to achieve the desired results. The advantages of a keyword approach are easy implementation and its conceptual simplicity for the user.

IJSER

- **Multi-attribute search:**

Multi-attribute search mechanisms [3, 4] use keywords, but also rely on other characteristics of the object being sought to be used as search keys. In the area of component retrieval, characteristics of components that can be used for retrieval are the class of the object (procedure, function, package, etc.), the number and types of parameters, the number of operations it supports, its domain of use, etc. An advantage to a multi-attribute search is that a component description contains more than just keyword information. The attributes taken together make up a classification scheme that provides more information than would be present in a pure keyword search. **A** disadvantage to a multi-attribute search is that the classification and subsequent storage location of a component defined **by** its attributes is left to the author and/or the library administrator, but different people will not necessarily classify the same component in the same way. **If** the user succeeds in filling in the same values, the search mechanism will be very precise, but unless some sort of partial matching function is used, recall of similar components will suffer.

- **Natural language interface:**

Historically, research in information retrieval has focused on textual document retrieval. It seems fitting to use natural language [5] queries to retrieve natural language data. The distinct advantage offered **by** this method lies in the ease of language query formulations **by** system users. In reusable component retrieval, the *higher* levels of language processing need to be applied. Of course these are the most difficult.

3.3 Formal Specification:

Many types of formal specification languages have been used to describe the semantics of software processes. Factors that contribute to their use as a means for component retrieval in the context of this research include .

1) A syntax or structure that is consistent with the structure of the underlying implementation language.

2) A means to execute the specification.

3) A facility for specifying generic components.

Three candidate specification formalisms are discussed here: predicate calculus, plan calculus, and algebraic formalisms.

Predicate calculus [6] is a specification language with a rigorous mathematical foundation. It is an *executable* specification language as well if you consider logic programming languages such as Prolog. The Plan Calculus [7] is a formalism developed for a system called the Programmer's apprentice. It combines the "representation properties of flowcharts, dataflow schemas, and abstract data types" to depict modules as a hierarchical graph structure. The theory of algebraic specifications is based on the notions of classical algebra in mathematics and on the concepts of abstract data types in computer science. It has its origins in the mid 1970's.

3.4 Case Based Retrieval:

The Case-Based approach enables reasoning about concepts, allowing the retrieval of "approximate" components. Literature [8] presented the reuse assistant, a hybrid approach to support the retrieval of software components from a library of object classes. The Reuse Assistant consists of two subsystems that follow two different approaches. Information retrieval techniques based on statistical methods, and knowledge-based techniques using some of the representation and indexing mechanisms found in case-based systems. Knowledge-based systems make use of a deep knowledge about the component design and implementation, and interrelate the components through a rich set of relationships expressing system architecture, design decisions and all the information needed to use and, more important, to reuse (adapt) the components. The major drawback of this approach comes from the need of a manual, high cost knowledge acquisition process, which handicaps the scalability and extensibility of these systems. On the other hand, automatic indexing systems make use of the readily available information, performing some kind of analysis on the text (and, sometimes, also on the code) associated with the components. Therefore, extendibility is granted because this process may be automatically applied to any new component added to the software library. So a hybrid approach was developed that consists of two subsystems to support the retrieval of software components from a library of objects classes. The Reuse Assistant consists of two subsystems: an IR module based on statistical methods, and a knowledge base, indexed with the techniques used in case-based (CB) systems. In this approach, every component is indexed using both techniques. In this way, the system offers a richer interface, where it is possible to pose queries in natural language (treated by the IR module), to incrementally fill a form specifying the user needs (access to the knowledge base), and to use any retrieved component as entry point to browse through the knowledge base.

3.5 External Classification:

External classification category includes all the approaches that represent the component by an external description. An automatic or a manual process can produce the description. It consists of Faceted Representation and vocabulary and classification using natural language description. Faceted classification approaches [9] for components retrieval consist of a collection of facets or classifications, which represent the type of information, which is relevant for identifying reusable components. Each facet has a name and an associated term-space called vocabulary, which is a collection of terms used to describe aspects of the facet. Faceted classification technique is powerful and gives good results. The disadvantage with this technique is that it requires manual indexing that can be expensive. Classification using natural language description [10] the system extracts lexical, syntactic and semantic information from the natural language description. The interpretation mechanism used for description analysis does not pretend to understand the meaning of the description, but it uses linguistic techniques to get automatically a component description that can be used to resolve the engineer queries. Classification techniques using natural language description are difficult to set up and specific to restricted domain use.

IJSER

3.6 Encoding component description:

It is a method used that can be used to mitigate

the classification and retrieval problem by encoding the semantic information [11, 12] of reusable components. The Software Resource Model (SRM) and Software Resource Diagram (SRD) are proposed as a solution to this problem, The SRM is used to make the description more precise and understandable by all classes of users, to remove inconsistencies among people. The SRD is the graphical representation of the SRM. The feasibity of SRM and SRD is tested through the laboratory experiment. Both have the consistency.

3.7 Optimization approach:

This brings out an application of optimization [13] for the planning of reusable software components (SCs). This present a model that selects a set of SCs that must be built, in order to lower development and adaptation costs. The key parameters involved in making this decision are as follows. For each software component (SC), the organization incurs a development cost, which comprises the programming cost to design and create the SC as well as the maintenance costs. In addition, each end-product has a demand. Finally, for each SC, some tasks result directly when the SC gets built, while others can be obtained by an appropriate transformation of the SC, at a certain adaptation cost.

3.8 Full Text Indexing:

Full-text indexing and retrieval of software components provided comparable precision but much better recall than controlled vocabulary indexing and retrieval of components. From the free-text end, research has aimed at making the matching more intelligent and less dependent on surface-level similarity, but keeping humans out of the loop—e.g. using associations between terms instead of term matching or identity, as in latent semantic analysis methods. Full-text indexing refers to the class of methods. Different methods use different selection mechanisms to restrict the set of eligible content indicators, and different weighting schemes. 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. 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. 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. It has several disadvantages:

- There is ambiguous nature of the command.
- Free text retrieval results in many irrelevant components.
- When a query is made search starts from beginning to end.
- Indexing cost is always there in free text.

3.9 Conversational case based reasoning:

In CCRM, components are represented as cases, a knowledge-intensive case-based reasoning (CBR) [14] method is adopted to explore context-based semantic similarities between users‟ query and stored components, and a conversational case-based reasoning (CCBR) technology is selected to acquire users‟ requirements interactively and incrementally. Case-Based Reasoning (CBR) is a problem solving method. The main idea underlying CBR is that when facing a new problem, we will search in our memory to find the most similar previous problem, and reuse the old solution to help solve the new problem. A CBR process can be divided into four phases: retrieve, reuse, revise and retain, Conversational case-based reasoning (CCBR) is an interactive form of case-based reasoning. CCBR has been probed in several application domains, for instance, the customer support domain9, and products or services selection in E-Commerce. A limitation of our method is its dependence on knowledge engineering.

3.10 Rough Fuzzy Approach:

Rough Fuzzy Approach [15**]** based on the rough–fuzzy sets, and a decision support tool based on this methodology is implemented. This methodology views retrieval and selecting components from a case based repository as a decision problem. If the case (component) retrieved is very similar to the problem being solved (selecting a component needed for reuse), adaptation and testing will be easy. The component match in several applications is classified using rough–fuzzy sets, based on the similarity of the attributes. A decision based on rough–fuzzy sets is applied to the history of a component’s use in different applications to determine its potential reuse in the current application.

Advantage of rough fuzzy approach:

- Classifies the components as adaptable to the given situation with membership values for the decisions.
- The use of rough–fuzzy sets increase the likelihood of finding the suitable components for reuse when exact matches are not available or are very few in number.

There is still a disadvantage for this as:
A software tool called RuFToo is developed as a decision support tool for component retrieval for reuse on Windows platform with MS-ACCESS as the back-end and Visual BASIC as the front-end to this purpose.

3.11 Genetic algorithm:

Genetic algorithm [16] addresses the issue of most appropriate component retrieval from a component repository. Appropriateness here is precision and quality. Two-step process is used for retrieval of appropriate component from repository. The initial step is Keyword based retrieval for finding all candidate components according to the user requirement. Keyword based search is used to narrow down the search space, which further improves initial population for Genetic Algorithm based second step. As Genetic Algorithms are applied to only extract candidate components, Thirty-two attributes of reusable component are considered for selecting the appropriate component. Here components are retrieved in two steps. The first step is retrieval based on user-defined keywords; the next step is optimizing the retrieved components using genetic algorithms. This approach enhances the chance of retrieving appropriate component from the repository that can be reused.

3.12 Intelligent Classification Scheme:

Intelligent classification scheme [17] used to retrieve the components. This method for the classification and retrieval of components from the reuse repository is divided into two phases. The first phase is component classification which is further divided into encoding and classifier discovery phase. The second phase explains the retrieval of component from the reuse repository. This classification scheme uses a genetic algorithm which evolves the small number of classifiers by dividing the set of available components stored into repository into certain subsets. It will result into the fast retrieval of correct components according to the requirements specified by the user. In the encoding phase components characteristics are encoded into the binary form that is understood by the genetic algorithm. The genetic algorithm processes the bit string and evolves the classifiers where each classifier holds the set of homogenous components. In the component retrieval phase, user will search for a specific component. First the user will enter the desired characteristics of a component 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 re-turned and higher the threshold value, exactly the components that matches user entered characteristics will get re-turned). 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.

3.13 Specification Based Approach:

Specification based approach [18] of business component retrieval based on specification matching to solve the software reuse of enterprise information system. Specification match relationships in two levels: business operation level and business component level. In business operation level, we use input business data types, output business data types and the taxonomy of business operations evaluate the similarity between business operations. In the business component level, we propose five specification matches between business components. To retrieval reusable business components, we propose the measure of similarity degrees to calculate the similarities between business components. Finally, a business component retrieval command like SQL is proposed to help user to retrieve approximate business components from component repository.
Comparing with the previous approaches, the proposed approach has the following characteristics. First, the proposed component model can describe both the static structure information about the interface and the dynamic behavior feature of a business component. The business type based on XML proposed can express the variable business data that can describe the variety of business operations. Second, it is a multi-layer matching mode that can enrich the semantic information of business component repository.

IJSER

3.14 Free Text and Faceted classification:

Free Text classification or keyword is used to classify and retrieve the components. 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. 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. 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. It has several disadvantages:

- There is ambiguous nature of the command.
- Free text retrieval results in many irrelevant components.
- When a query is made search starts from beginning to end.
- Indexing cost is always there in free text.

3.15 Behavioral and structural classification:

Henen Ben Khalifa [19] suggested a technique called as structural classification in 2008 for the retrieval of component that includes all the approaches that represent the component by extracting its structure. This technique selects candidates not on the basis of their function but rather on the basis of their structure. Mili A. et al mentioned, two distinct families of criteria that we can use to select candidate components in a software library: *functional criteria*, which seek to identify components that have the same functional properties as the query (*act like* the query); *structural criteria*, which seek to identify components that have the same structure as possible solutions to the query (*look like* the query). Typically functional criteria are best adapted to black box reuse, whereby components are used verbatim (without modification). Structural criteria are best adapted to white box reuse, whereby components are used after modification – although it is not uncommon that functional criteria are used. There are two methods that are used to retrieve component in structural classification, that are signature matching and specification matching. Specification matching primarily relies upon matching predicates in logic. Predicates are expressions written in a logic using terms and formula symbols. B or OCL can be used to write the components specifications and to specify the query. The engineer query is a specification of the component that the engineer needs. The component retrieval system uses proof theorem method to match the query with the components library specifications. Signature matching approach primarily relies upon type matching and type transformation. Classes are represented by a multi-set of feature signature. Signature matching approach assumes that the set of allowable signatures in the system is known. There are several disadvantages with structural classification such as:

- It is only aimed at white box reuse, so it only searches for approximate retrieval of the components.
- Till date these methods are mainly in laboratory use.
- Proof theorem that is required to match in library is very complex.
- Signature matching technique is useful when the application engineer knows a partial or a complete signature definition of the needed component.

Qualid Khayati [19] described the **behavioral retrieval technique** to retrieve the component from repository. This focuses mainly on the behavior of the component. Behavioral retrieval works by exploiting the executability of software components. Programs are executed using components, and the responses of components are recorded. Retrieval is achieved by selecting those components whose responses (with respect to the program) are closest to a pre-determined set of desired responses. This idea was originally called ``behavioral matching". In general, Behavioral matching executes each component with input data in order to retrieve the components that present the expected behavior. In general, Behavioral matching executes each component with input data in order to retrieve the components that present the expected behavior. There are some disadvantages with this also such as:

- It uses randomly chosen “samples” to execute all operations in software library with a signature that matches the required one.
- Since the developer has to calculate the expected result by hand, the approach clearly has some similarity to what we call black-box testing today.

IJSER

3.15 Attribute value:

Attribute value classification uses a set of attribute to classify components. For example, a book has many attributes such as the author, the publisher, and a unique ISBN Number and a classification code in the decimal system. Depending upon who wants information about a book, the attribute can be concerned with the number of pages, the size of the paper used, the type of print face, the publishing date etc.
The attributes relating to a book can be:

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

This technique has a disadvantage that is many irrelevant components can be retrieved.

4. Conclusion and Future work:

This paper aims at the components based on facets description, draws supports from the ideas of tree matching,attributes,signature matching, specification matching, hypertext, Browsing technique.These retrieval methods for large scale component library for retrieval will be on one hand meet various needs to retrieve, on the other hand, they can ensure the efficiency of the retrieval. In order to further improve the retrieval algorithm precision and recall rates, and how to extend the term dictionary; how to expanded retrieval condition in case of non-modify the program; How to design good retrieval interface and efficient retrieval platform for component reuse are next steps of retrieving the key issues.

REFERENCES:

[1] Meyer, Bertrand, “Object-Oriented Software Construction”, Prentice Hall, **1988.**
[2] Salton, G. and McGill, M., “Introduction to Modern Information Retrieval”, McGraw-Hill, **1983.**
[3] Prieto-Diaz, Ruben, **“** Software Classification Scheme”, Ph.D. Dissertation, University of California, Irvine, California, **1985.**
[4] Brown, James **C.,** Lee, Taejae, and Werth, John, "Experimental Evaluation of a Reusability-Oriented Parallel Programming Environment", IEEE Transactions on Software Engineering, v. **16, pp.** 111-120, February **1990.**
[5] Salton, G. and McGill, M., “Introduction to Modern Information Retrieval”, McGraw-Hill, **1983.**
[6] Rollins, Eugene **J.,** and Wing, Jeanette M., "Specifications as Search Keys for SW Libraries: **A** Case Study using Lambda Prolog", **CMU-CS-**90-159, Carnegie Mellon University, 26 September **1990.**
[7] Rich, Charles, and Waters, Richard C.” The Programmer's Apprentice”, Addison-Wesley, **1990.**
[8] Carmen fernandez-Chamizo, Pedro A, Gonzalez-Calero, Luis Hernandez-Yarez, And Alvaro Urech-BAQUI,”case based retrieval of components”**1991.**
[9] R. Prieto-Diaz and P. Freeman, “Classifying Software for Reusability”. IEEE Software, 4(1), pages 6-16, **1987.**
[10] M. R. Girardi, B. Ibrahim, “A software reuse system based on natural language specifications”, Proceeding of International Conference on Computing and Information (ICCI’93), Sudbury, Ontario, Canada, pages 507-511, **1993.**
[11] W.B, & Fox C.J. Sixteen questions about software reuse. Communication of the ACM **1995**.
[12] C.W. Software reuse. ACM computing surveys ,24(2). 131-183**,1992.**
[13] B. Boehm, Managing software productivity and reuse, Computer 111–113,**1999**
[14] Aamodt, A. and E. Plaza, “Case-Based Reasoning: Foundational Issue, Methodological Variations, and System Approaches”. AI Communications, 7(1): p. 39-59 **1994.**
[15] D., Prade, H.,” Rough–fuzzy sets and fuzzy–rough sets”. Int. J. Gen. Syst. 17 (2–3), 191–209, **1990.**
[16] Jun-Jang Jeng Betty H. C. Cheng,” Specification Matching for Software Reuse: A Foundation”, ACM, pp 97-105,**1995.**
[17] V. Maxville, J. Armarego, C.P. Lam, “Intelligent Component Selection”, 28th Annual International Computer Software and Applications Conference, COMPSAC, pp 244-249, **2004.**
[18] “A Specification-Based Approach for Retrieval of Reusable Business Component for Software Reuse” by Meng Fanchao, Zhan Dechan, and Xu Xiaofei,**2006.**
[19] “ A Behavioral and Structural Components Retrieval Technique for Software Reuse” by Hanen Ben Khalifa, Oualid Khayati, Henda Hajjami Ben Ghezala,**2008.**

IJSER