Development of a Software Repository for the Precise Search and Exact Retrieval of the Components

Thesis submitted in the partial fulfillment of the requirements for the award of degree of

Master of Engineering

In

Software Engineering

Submitted By:

Amandeep Bakshi

(801131001)

Under the supervision of:

Dr. (Mrs.) Seema Bawa

Professor and Dean of Student Affairs

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA-147004, INDIA

June 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*Development of a software repository for the precise search and exact retrieval of components*", in partial fulfillment of the requirement for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala , is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa and refers other research works which are duly listed in the reference section.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other University.

Amandeep
12/July 2013

(Amandeep Bakshi)

(801131001)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Seema Bawa
12/July/2013

(Dr. Seema Bawa)

Professor and Dean of Student Affairs
Computer Science and Engineering Department, Thapar University,
Patiala- 147004.

Countersigned by

Dr. Maninder Singh
Head
Computer Science and Engineering Department
Thapar University, Patiala

Dr. S.K. Mohapatra
Dean(Academic affairs)
Thapar University
Patiala

Acknowledgement

A journey is easier when traveled together. Interdependence is certainly more valuable than independence. This thesis is the result of work carried out during the final year of my course whereby I have been accompanied and supported by many people. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

I would like to express my sincere appreciation to my supervisor Dr. Seema Bawa, Professor & Dean of student affairs, Computer Science & Engineering Department for her excellent guidance, pillar of support and encouragement right from beginning of this course. No amount of words can adequately express the debt for her kind support, motivation and inspiration that triggered me for the thesis work

I am also thankful to all the faculty and staff members of the Computer Science and Engineering Department for providing me all the facilities required for the completion of this work.

No thesis could be written without being influenced by the thoughts of others. I would like to thank my classmates specially Jagdeep Singh who were always there at the hour of the need and provided with all the help and support, which I needed and helped me with his kind suggestions.

I offer my deepest gratitude to my all family members especially my father and mother for their love, encouragement, motivation and for their confidence in me.

At last but not the least I would like to thank "**The Creator of Destinies**" for not letting me down at the time of crises and showing me the silver lining in the dark clouds.

Amandeep

Amandeep Bakshi

(801131001)

Abstract

The recent progress in the component based engineering has offered many facilities in the field of software reuse. However, this advancement has also brought the challenge such as designing repositories and developing efficient search of software components. But the problem that is generally encountered is the efficient search and the exact retrieval of the software components from the software repositories as they are not properly represented yet.

There are many softwares and websites which provide for the retrieval of reusable components, but, the search is not precise and the retrieval is not that exact. So, the system proposes to make the search as concise and precise as possible and also facilitate the users with various retrieval methods, out of which they can select the one they deem fit for their purpose.

The user should be able to perform search based on certain fixed categories by using different retrieval techniques i.e. Keyword based retrieval, signature based retrieval and operational semantic based retrieval. The basic comparison between these techniques and the test cases are designed on the basis of precision and recall should be done for improving the effective search.

Thus, this dissertation deals with the construction of the software repository, construction of the operational semantic keyword based retrieval technique for evaluating the best results and at the end comparison between the techniques for retrieval of the software components is made on the basis of precision and recall.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this chapter is to introduce the software reuse area to the reader. This chapter also presents software reuse repository and the various component storage and retrieval techniques. In addition to that, this chapter contains the assessment criteria for the component storage and retrieval methods.

1.1 Software Reuse

Software reuse is basically consists of making use of any existing information, component or product when designing and implementing a new system or a product. It means using a segment of source code that can be used again to add new functionalities with some modification. Replication of an entire software program does not count as a reuse. Reuse of assets is dependent upon both the similarities and differences between the applications in which the component is being used [2]. Many organizations already practice a limited form of reuse for example most developers have repositories of software components that they have developed in previous projects or they use standard libraries which are available with many programming languages [1]. About 30% of the cases, it is a very ad-hoc method of reuse, and it will work very well on a small scale and it will not be suitable for entire organizations [3].

Instead, businesses need to implement a systematic reuse program in order to gain the full advantages of reuse. Reuse can be achieved through different modes. *Compositional* reuse involves constructing new software products by assembling existing reusable assets. *Generative* reuse involves the use of application generators to build new applications from high level descriptions. So, software engineering has been more focused on original development but it is now recognized that to achieve better software, more quickly and at lower cost, we need to adopt a design process that is based on systematic software reuse.

1.2 Software Reuse Repository

Software Reuse Repository is simply a component library which stores the reusable components and must have the characterization of the assets that are included within. In order to make a effective use of a software repository, a reuser must have a clear understanding of its contents, so as to determine that whether his needs are likely to be met by the library [4]. So, Software Repositories must be designed to meet the evolving and dynamic needs of software development organizations [5]. Repositories are used as mechanisms to store, search and retrieve components. But finding and reusing appropriate software components is often very challenging particularly when faced with a large collection of components and little documentation about how they can and should be used. Many software component repositories have been developed often extending the approaches used for software libraries. Example includes CodeFinder Repository [6].

1.3 Software Reusable Components

The definition of a software reusable component is "any component that is specifically developed to be used and is actually used in more than one context" [7]. This does not just include code, other products from the system lifecycle can also be reused such as specifications, requirements and designs. Components in this case can be taken to include all potentially reusable products of the system lifecycle including code, documentation, design, requirements, architecture etc as following [8].

- **Project Plans:** The basic structure and much of the content of a software project plan can be reused from project to project. This reduces time to develop plan and uncertainty associated with establishing schedules, risk analysis and other features.
- **Cost Estimate:** Often different projects have similar function; it may be possible to reuse estimates for that function with little or no modification.
- **Architecture:** There are relatively little distinct programs and data architecture, even when different application domains are considered.

- **Designs:** Architectural, data interface, and procedural designs developed using conventional methods are candidate for reuse. More commonly, system and object designs are reused.
- **Source Code:** Verified program components written in compatible programming language is candidate for reuse.
- **Human Interfaces:** Probably the most widely reused software artifact is Graphical User Interface, which is commonly reused.
- **Data:** Among the most commonly reused artifacts, data encompasses internal tables, lists, and record structures as well as files and complete databases.
- **Test cases:** Whenever a design or code component is to reuse, the relevant test cases are also reused.

These artifacts constitute a reuse library. These libraries must contain not only reusable components but are also expected to provide certain types of services like storage, searching, inspecting and retrieval of artifacts from different application domains, and of varying granularity and abstraction, loading, linking and invoking of stored artifacts, specifying artifact relationships, etc. The major problems in software libraries are classification and retrieval of the components from these libraries.

1.4 Characteristics of Software Components

Certain software artifacts have been reused these artifacts usually share a number of characteristics, to increase reusability [8].

- **Expressive:** They are of general utility and of adequate level of abstraction, so that they could be used in many different contexts, and are applicable to variety of problem areas.
- **Definite:** The artifacts should be constructed and documented describing their purpose, their capabilities and limitations are easily identifiable, interfaces, required resources, external dependencies and operational environments are specified, and all other requirements are explicit and well defined.
- **Transferable:** The artifacts should be easily transferable from one environment to different environment or domain.

- **Additive:** The artifacts should be easily fitted to new application without requiring enormous modifications or causing adverse side effects.
- **Formal:** Reusable artifacts should be described using formal and semiformal notations so that the verification of artifacts should be easy.
- **Machine Representable:** Those of the artifacts which can be described in terms of computationally determined attribute values, which can easily be decomposed into machine represent able parts, which can be accessed, analyzed, manipulated and possibly modified by computer-based processes, have a clear potential for becoming part of a flexible reuse library; those artifacts can be easily searched for, retrieved, interpreted, altered and finally integrated into larger system.
- **Self-contained:** Reusable artifacts which embody a single idea are easier to understand, they have less dependency on external factors, whether environmental or implementation, they have interfaces which are simple to use, they are easier to extend, adapt and maintain.
- **Language Independent:** The artifacts should not be embedded with implementation details in reusable artifacts.

1.5 Component Storage and Retrieval Techniques

Consider a large university library. Tens of thousands of books, periodicals, and other information resources are available for use. But to access these resources, a categorization scheme must be developed. To navigate this large volume of information, librarians have defined a classification scheme that includes a library of congress classification code, keywords, author names and other index entries. All enable the user to find the needed resource quickly and easily. Similarly in component library, tens of thousands components are stored. How users can finds out the appropriate one. The solution to this problem is classifying the software libraries in a particular order and use effective retrieval technique.

Another Problem rises with the existence of these software assets or components are the presence of a storage structure or a repository that would contain these components and also provide for an easy search and retrieval of the same. The repository has to be

constructed and maintained in order to maximize the use of reusable components. In addition to construction of the repository, some retrieval technique also needs to be implemented that utilizes minimal repository structure to effectively support the process of finding software components. Once the repository has been constructed then the information that needs to be stored in it should also be structured in a way such that it should be digitalized, normalized and archived.

After the assets have been properly stored in the repository, the next problem that arises is finding of the appropriate asset by the user and proper query formation for the same purpose. This has to be tackled by applying proper classification schemes which also form an important part of the retrieval process. The classification is important in order to tackle the search problem that arises due to difference in the users thought process and the final query formation as they help in organizing the assets in the repository in an organized manner.

The retrieval of any component is measured according to abstract performance measures such as precision and recall. Hence, the process that needs to be followed for retrieval should guarantee high performance measures. A number of retrieval techniques are in place and the most appropriate one can be chosen according to the requirements and the available resources. Hence, it reduces the effort, time and other resources that go into building the components again and again.

1.5.1 Assessment Criteria for Storage and Retrieval Methods

The basic criteria for comparing the various storage and retrieval methods are [9]:

- **Precision**: The precision of a retrieval algorithm is the ratio of the relevant retrieved assets over the total number of retrieved assets; this number ranges between 0 and 1. Under the hypothesis that all library assets are visited (example, exhaustive navigation) we get perfect precision (=1) whenever, the matching condition logically implies the relevance criterion. This can be achieved in particular by letting the matching condition be false, which means that no assets are returned (hence no irrelevant assets are returned). When concrete data are not available to quantify

precision, we may instead assign values in a discrete five value rating [very low (VL), low (L), medium (M), high (H), very high (VH)] where, VH refers to perfect precision and VL refers to poor precision.

- **Recall**: The recall of a retrieval algorithm is the ratio of the relevant retrieved assets over the total number of relevant assets in the library; this number ranges between 0 and 1. Under the hypothesis that all library assets are visited (example, exhaustive navigation) we get perfect recall (=1) whenever, the matching condition logically implies the relevance criterion. This can be achieved in particular by letting the matching condition be false, which means that no assets are returned (hence no irrelevant assets are returned). When concrete data are not available to quantify recall, we may instead assign values in a discrete five value rating [very low (VL), low (L), medium (M), high (H), very high (VH)] where, VH refers to perfect precision and VL refers to poor precision.

The aim of any search program is to maximize precision and recall, so as to provide the effective retrieval.

1.5.2 Characterizing of Storage and Retrieval Methods

Methods for the storage and retrieval of software components in software repositories abound in the literature. In order to understand the basis for classification, we characterize each method by a number of orthogonal attributes. These attributes are discussed as follows [9].

I. **Nature of asset:** The most important attribute of a software library is the nature of the assets that are stored in it. Asset can be a source code, executable code, specifications, designs, test data, documentation etc.

II. **Scope of library:** Another attribute of a software library is the scope of the library means whether the library is expected to be used within a single project, within an organization or on a larger scale.

III. **Query representation:** In software library we can represent the query in a formal functional specification, a signature specification, a behavioral sample, a natural language query or by the set of keywords.

IV. **Asset representation:** The representation of a assets is very important feature of a library, not only because it dictates what form user queries take, but also because it determines how retrieval is performed.

V. **Storage structure:** The most common logical storage structure in software libraries is no structure at all because software assets are stored side by side with no ordering between them .While the components are ordered by their identifying keys in traditional database systems because it is difficult to define a general key that can be used to order software assets in a meaningful way.

VI. **Navigation scheme:** In case of a flat storage structures, the only possible pattern of navigation is brute force exhaustive search of all the entries. Whenever the assets are arranged on some non-trivial structure, this structure can be used to help orient the search towards those assets that are most likely to satisfy the query and steer it clear from those that are known to be irrelevant or are thought unlikely to be relevant.

VII. **Retrieval goal:** Basically the goal of a retrieval operation is to find the correct one component with respect to a given query. If the retrieval operation fails to find the exact candidate component, one may want to perform another retrieval operation with the lesser ambition of finding the component that approximately fulfills the query.

VIII. **Relevance criterion:** The Relevance criterion defines under what condition a library asset is considered o be relevant for the submitted query with respect to the predefined retrieval goal.

IX. **Matching Criterion:** The matching criterion is the condition that we choose to check between the submitted query and a candidate library asset to decide whether the asset is relevant. Ideally the matching criterion should be equivalent to the relevance criterion, but it is not always so, if the representation of the asset or asset surrogate is too abstract then these two criteria may differ significantly.

1.6 Thesis Organization

The **first chapter** briefly describes the background study and introduces the motivation behind the development of the project and the organization of the whole thesis.

The **second chapter** deals with the literature survey; this chapter theorizes the basis of the implementation of the system.

The **third chapter** defines the problem statement; it illustrates the difference between the already existing system and the proposed system.

The **fourth chapter** gives details about the implementation of the interface .It defines the basic feature of the system along with the construction of the software repository and established the various methods of search and retrieval of the software components.

The **fifth chapter** deals with the testing details and test cases obtained by the system and experimental results obtained of the system on the basis of precision and recall ratio.

The **sixth chapter** reflects upon the conclusion and future scope of the thesis report. And at last are the references and papers published.

Chapter 2

Literature Survey

The purpose of this chapter is to give the user an overview of the existing techniques, methods for the effective search and the retrieval of the components from the software reuse repositories.

2.1 Software Repository

As libraries of reusable software components continue to grow, the issue of retrieving components from software components repositories has captured the attention of the software reuse community [6]. The storage and retrieval of reusable assets in a software repository has been the subject of active research in the past. Reusable assets includes program specifications, source code, object codes, documentations and the UML specifications that are produced in a project and can be later reused. However, finding a components and reusing the appropriate software components is often very challenging, particularly when faced with a large collection of components and little documentation is there about how they can be reused. This is a particular issue for end users of component-based systems who want to tailor and extend their environment, but have limited understanding of component functionality and implementation [11]. Many software components repositories have been developed, often extending the approaches used for software repositories. Hence, it is becoming difficult to retrieve relevant components. This motivates researchers to look for retrieval techniques that give efficient guidance for locating and identifying appropriate software components that satisfies user queries [12]. Various retrieval techniques are enumerated classification, facets, frame-based classification; free-text indexing and relational databases have been employed to address the problem of finding relevant components [6]. But issues involving how effective repositories are built, populated and evolved to meet the changing needs of software developers have received considerably a less attention. Thus, this thesis describes a research on effective search and retrieval of software components from software

repositories. This Literature review aims at summarizing the current state of the art in software reuse repositories research by proposing answers to the following questions:

(i) What are the requirements of a software reuse repository?
(ii) How the software repository must be designed?
(iii) What are the various approaches followed for constructing effective repository?
(iv) What are the main challenges faced while searching the components?
(v) What are the main challenges faced while retrieving the components?

2.1.1 Creating a Software Repository

Any repository may or may not supporting reuse, will be constructed by following a common procedure for its development [10]:

- **Making the Business Case:** The case for a repository must be made by the institution or community that will own and sustain it. In justifying a repository it is critical to work out a case that best aligns with the priorities of the institution. A carefully prepared case to senior management will highlight the appropriate advantages of the repository to the institution, will detail expected expenditure over a number of years and will emphasize that the payoff is not measured in financial terms.
- **Defining the Purpose of the Repository**: Repositories can have a wide variety of uses, hence, repository services should be developed with a clear idea of the purpose of the repository in mind.
- **Defining Repository Services:** There are many different services that can be provided by a repository, but, a particular repository should offer a particular service only. For example, a repository developed by a business organization for the purpose of payroll management of employees should provide with data and components related to this purpose only.
- **Choosing Repository Software**: Based on the needs and services of the repository, institutions will then want to assess the available software platforms

and choose the one, most appropriate to the organization. It may be open source or commercial.

- **Developing Repository Policies:** Three policy areas need to be addressed in relation to repositories: Collection, Management and Access.
- **Marketing the Repository**: Populating the repository is one of the greatest challenges for repository managers. The ongoing promotion of the repository is key to ensure its visibility and success. A variety of methods can be used to market the repository so that, it is used by a wider audience.

2.1.2 Existing repositories for the effective search and retrieval of components

The existing repositories for the effective search and retrieval of components are as follow:

2.1.2.1 The CodeFinder-PEEL Repository

Scott Henninger outlines an Evolutionary approach [6] in 1996 for constructing effective software libraries which is demonstrated through CodeFinder-PEEL repository (Parse and Extract Emacs Lisp) which extracts components from text files and indexes these components through a combination of automatic extraction. The CodeFinder system is a prototype retrieval tool that uses a combination of retrieval techniques to help users for finding reusable software components. This repository was designed to investigate the cost/benefit tradeoffs.

Components are extracted through PEEL and translate these components into a CodeFinder representation. Once, components are translated, seeding a repository is necessary for creating the component representations by indexing them with phrases and key terms. PEEL is a reengineering tool that translates Emacs Lisp files into the individuals reusable components representing into the frame based language named kandor which is used by CodeFinder for indexing the components and create a frame based hierarchy of retrieval concepts. PEEL extracts source code definitions of functions, constants, variables and macros from a source code file. Then this information is

extracted and translated into kandor objects. Kandor representation can be viewed as a set of attributes which contain information about the component.

This approach basically shows how repositories can be evolutionary constructed with minimal up-front effort.

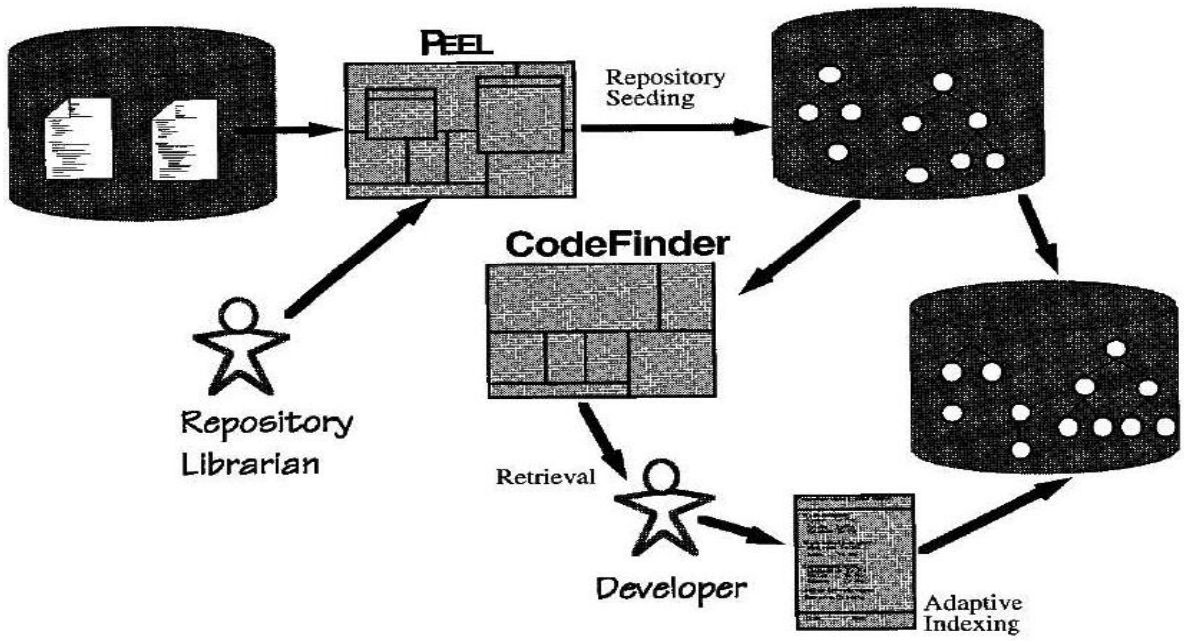

Figure 2.1: Code Finder PEEL Repository [6]

2.1.2.2 CodeBroker

Yunwen Ye discusses an active and adaptive reuse repository system in 2001 [13] to address the various challenges and demonstrates the feasibility of implementation through a prototype „Code Broker" system to assist the java developers in reusing various classes and methods. The Code Broker automatically locates the relevant software reusable components and delivers these right components into the development environment. The architecture of Code Broker consist of different software agents i.e. listener, fetcher and presenter. Listener and Presenter are the interface agents which connect the software developers to the backend information agent i.e. fetcher. Listener captures the software developers needs for reusing the components and formulates these queries autonomously. Fetcher retrieves these relevant reusable components based on the concept similarity and constraints. Fetcher is a backend information agent which retrieves

the components from the repository. Presenter uses a user profile for each software developers to adapt the components retrieved by the Fetcher to the developer's knowledge level. CodeBroker is currently designed to promote reuse in the phase of coding.

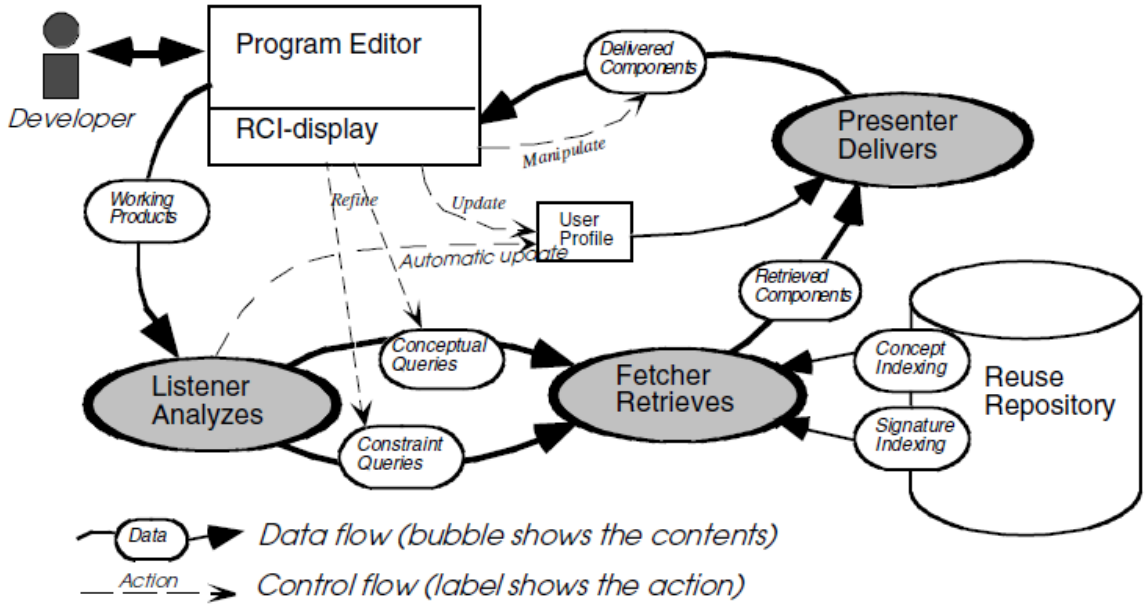

Figure 2.2: The architecture of CodeBroker [13]

For Example A software developer generate random number between 2 integers, before he or she implements it, it must be indicated in doc comment. Listener extracts the contents: create a random number between two limits. Then this concept is passed to the fetcher and presenter, which present those components whose functional descriptions match the query. If the developer cannot find the needed component immediately, he or she may proceed to declare the signature of the program. When the developer types the left bracket{ (just before the cursor), Listener recognizes it as the end of a signature definition. Listerner thus creates a constraint query: int x int -> int and passes to the fetcher. This Constraint compatibility is determined by signature matching.

2.1.2.3 Aspect-based component repository

John Grundy describes a software component repository [14] in 2001 which uses the concept of "aspects" to index and query the various software reusable components used

for automatically generate a high level indexing based on the systemic characteristics. A key aim of this approach is to make it easier for developers and for the end users to formulate the high level queries for components and have access to high level information about the retrieved components. Thus every component is added into the repository with its aspect details so that component is easily indexed and queried by using the aspect details for efficiently retrieval.

This technique is primarily a facet-based approach where users query for components based on the particular systemic aspects i.e. attributes of components.

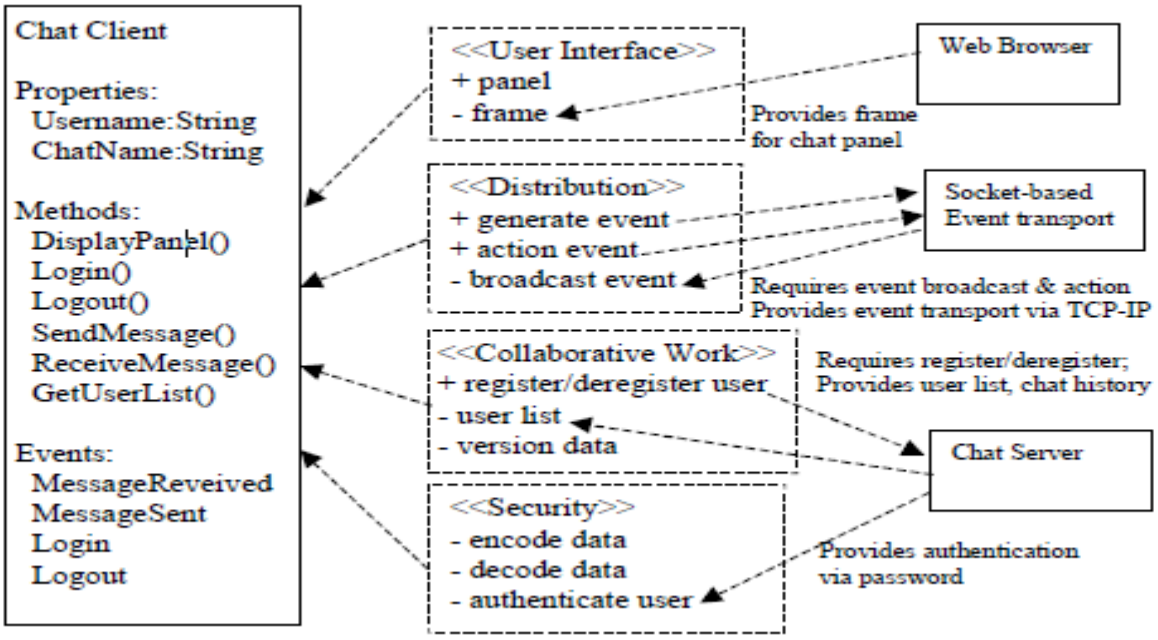

Figure 2.3: Concept of component aspect [14]

This Aspect based repository uses the concept of high-level, systemic aspects of a software application to characterize component provided and the required services. A key aim of our component repository approach is to make it easier for developers and end users to formulate high –level quires for components and have access to high level information about components retrieved. Thus, every component added to the repository needs to have its aspect details queried and be indexed by these aspect details to facilitate retrieval.

2.1.2.4 CRECOR Component Repository

Jihyun Lee, Jinsam Kim and Gyu-Sang Shin describes the component repository [15] in 2003 which supports for facilitating EJB as a reusable software component in component based software development CRECOR(Component Repository for facilitating EJB Component Reuse) is developed to store components and support reuse activities which also provides the capability to reuse the pre built EJB software components with GUI. We analyze the EJB component based on its deployment descriptor and extract component structure and organizing elements such as bean name, interface name and method signatures. Various Reuse facilitating activities are component analysis, component adaption, component deployment, and component test and assembly are described in this approach so that components can be efficiently reused.

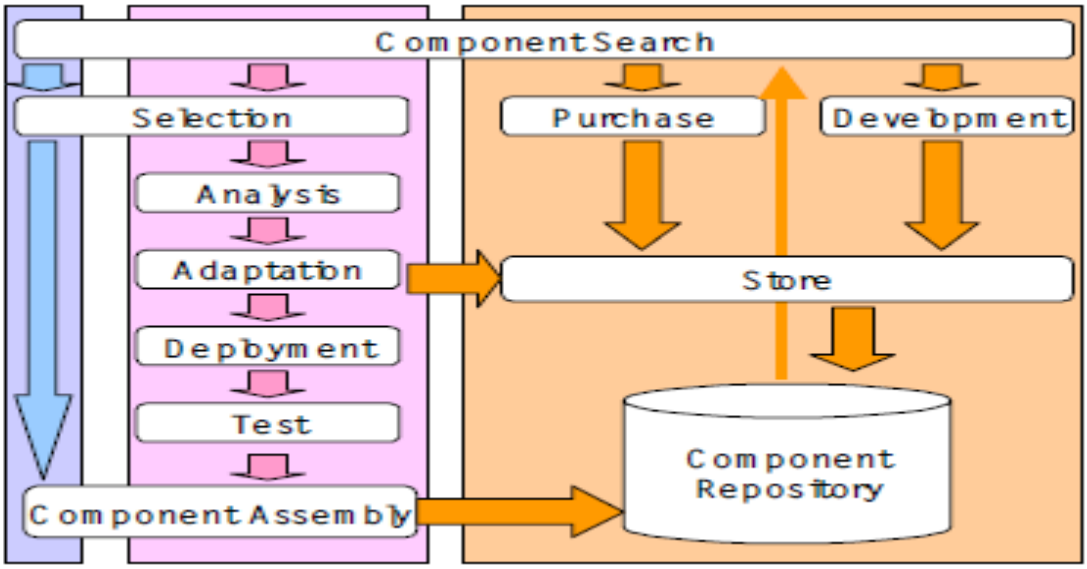

Figure 2.4: Component Based development process using EJB [15]

2.1.3 Problem faced in construction of software repository

We are faced with a number of problems when it comes to repository construction for reusable components. Some of them are as follow:

I. The repository when constructed has to have in addition to the basic structure, the system that facilitates easy search and retrieval of the components, which will increase the cost of construction.

II. A repository created with minimal retrieval structures, like those created when we use mechanisms like PEEL, may suffer from incomplete and inconsistent indexing, making it difficult for keyword-matching algorithms to retrieve relevant information.

III. When automatic parsing is applied in order to prepare the index for the assets in the repository, the vocabulary becomes limited moreover; there are no means to extract information of the assets other than the text files.

IV. The user access to the listed index also extends to the editing by the user, this would only add to the list of words in the vocabulary and also the search criterion becomes vague.

2.2 Component Search and Retrieval process

A search and retrieval technique is concerned with the comparison of the representation of a query and the representations of component for the purpose of identifying and or retrieving those documents that are relevant to that query. A retrieval technique is not directly concerned with the way of actually representing the component and queries, but different ways of representation may imply using different retrieval techniques. In this sense representations and retrieval techniques are interrelated. In principle, information storage and retrieval is simple. Suppose there is a store of documents and a person (user of the store) formulates a question (request or query) to which the answer is a set of documents satisfying the information need expressed by his question. User can obtain the set by reading all the documents in the store, retaining the relevant documents and discarding all the others. In a sense, this constitutes 'perfect' retrieval. This solution is obviously impracticable. A user either does not have the time or does not wish to spend the time reading the entire document collection, apart from the fact that it may be physically impossible for the user to do so. Often also the term retrieval model is used.

In the literature there exists a lot of confusion about the difference between model and techniques. In fact both terms are frequently used without, that it is really explained why. If we want to make some difference after all we could roughly say that a retrieval technique is mostly concerned with the matching of the document descriptors and the query representations. Whereas a retrieval model is going a little beyond this, also taking into account the way of representing the documents and the indexing process. Intellectually it is possible for a human to establish the relevance of a document to a query. For a computer to do this we need to construct a model within which relevance decisions can be quantified. It is interesting to note that most research in information retrieval can be shown to have been concerned with different aspects of such a model. There exist a great number of retrieval models or techniques providing different approaches to the way of locating information.

2.2.1 Existing Search and retrieval techniques

Many Search and retrieval techniques are available for the effective retrieval of the components from software repository which are as follows:

2.2.1.1 Component Storage and Retrieval method

A.Mili , R.Mili and R.T.Mittermeir presents a survey on methods of storage and retrieval of software assets in software component libraries in 1998 [9]. They classify storage and retrieval methods on the basis of which software libraries can be characterized. They define the assessment criteria which includes technical, managerial and human factors (precision, recall and coverage ratio factors) for the effective retrieval of components. On the basis of these assessment criteria, they classify the software libraries which include informational retrieval method, descriptive method, operational semantic methods, devotional semantic method, topological method and structure method.

2.2.1.2 SOM and GSOM Techniques

Ronaldo C.Veras and Silvio R.L.Meira both discussed and compared the two clustering techniques [16] in 2007 namely Self-Organizing maps (SOM) and Growing Hierarquical

SOM (GHSOM) for clustering a repository of classes from a java API for building mobile system to provide a more visualization of the software components and to refine more search by grouping together similar components. Search and Retrieval of software components can be developed using these clustering techniques. Suppose we have a software development environment in which we need to search for code in the software repository. In this case, the utilization of clustering algorithms could be useful to software developers because they could refine their research results and obtain other components based on the similarities. The process is shown in Figure below. In this process, the developer needs to express his query by using features that adequately represent a software component via query terms.

In the Second step, the query terms are used to carry out the search in the repository and as a result, certain components are selected. These techniques improve the component retrieval process.

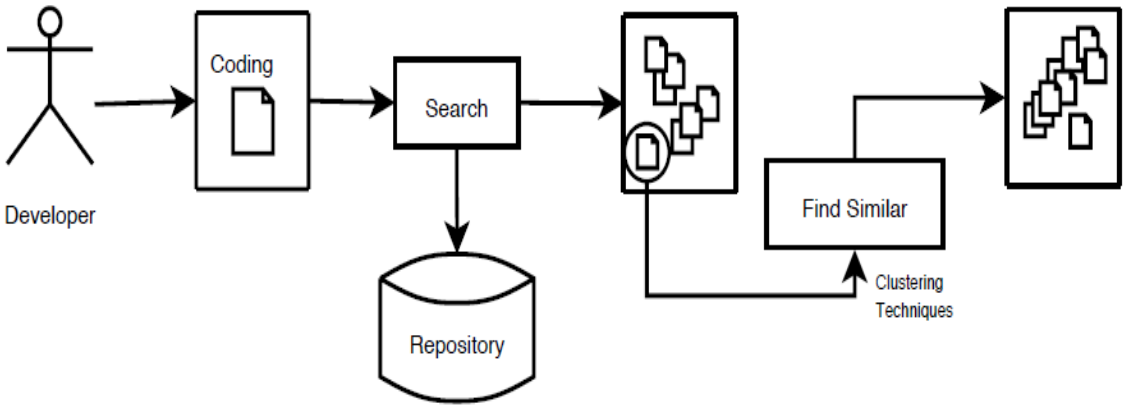

Figure 2.5: The use of clustering techniques in the software development cycle [16]

2.2.1.3 Structural and behavioral Techniques

Hanen Ben Khalifa and Qualid Khayati purpose structural and behavioral techniques [17] in 2008 for making the more efficient retrieval process by combining formal and semi formal specification for the heterogeneity of the software repositories. They purpose another classification for the retrieval of components that classifies the components into three different dimensional spaces. The first axis describes the interaction by using query or browsing between the user and the retrieval system. The second axis focus on the

description model for the software components. The last axis distinguishes between the structural and behavioral method. The behavioral based retrieval approaches are based on the notion of exploiting the executability of software components to classify them. Testing the components with different arguments calling their functions yields dynamic responses, which are collected. This collection is called the component behavior. An ordering on behaviors is then used to classify components and to search through the library of components. The programs used to produce the components behavior tries to call a subset or all the functions of the component and recover the results. If the program calls functions that do not exist in the component, the components will ignore the call. The person calling some specific function will enter a specific query and this query will be plugged to all components of the library to test the components behavior. The components that respond to the searched behavior will be selected and presented to the user [18, 19].

2.2.1.4 Hypertext Technique

A hypertext consists of a set of interconnected units of textual information such a unit of information is called a node. The connections between the nodes are called links. The starting point of a link is called the anchor or the parent node. The ending point is called the destination or the child node usually links are one directional but it is also possible to define two dimensional links. Nodes can be anchors and destinations of more than one link. By means of this set of nodes and links a structure is imposed on the collection of data a browsing feature which has been classified as a retrieval technique earlier permits the user to wander through the data thus providing a natural way of accessing the collection of data. Nodes can be of any nature, they could for example be titles of sections or chapters in a book or they could be bibliographical citations names of figures tables etc. Links are pointers from one node to another. They could be references to sections figures or table's citations etc. The actual information is stored in a database and the nodes of the hypertext correspond to pieces of the information. Only if a user finds a node interesting enough the actual information is retrieved [20,21].

A hypertext is thus a network structure on top of the actual information in the document collection. Following the links in the network through browsing there may be several different ways to reach the same node i.e. there are several different paths to the same node from a certain starting point. Following one directional links usually means more refinement. At a certain point the user might find out he has gone into too much detail [21]. It is then possible to back track along the path one has taken to that particular node. It is usually not possible to go back to a different anchor node than the one the user came from, if the node containing the requested information is found the search can be ended. The path from a starting point to the final destination node is called a search path. Figure 2.6 presents a simple example of a hypertext structure. The nodes are assumed to contain some kind of textual information the links represent some kind of relationship between these pieces of textual information. For the sake of this example it is not necessary to actually specify the nature of the nodes and links.

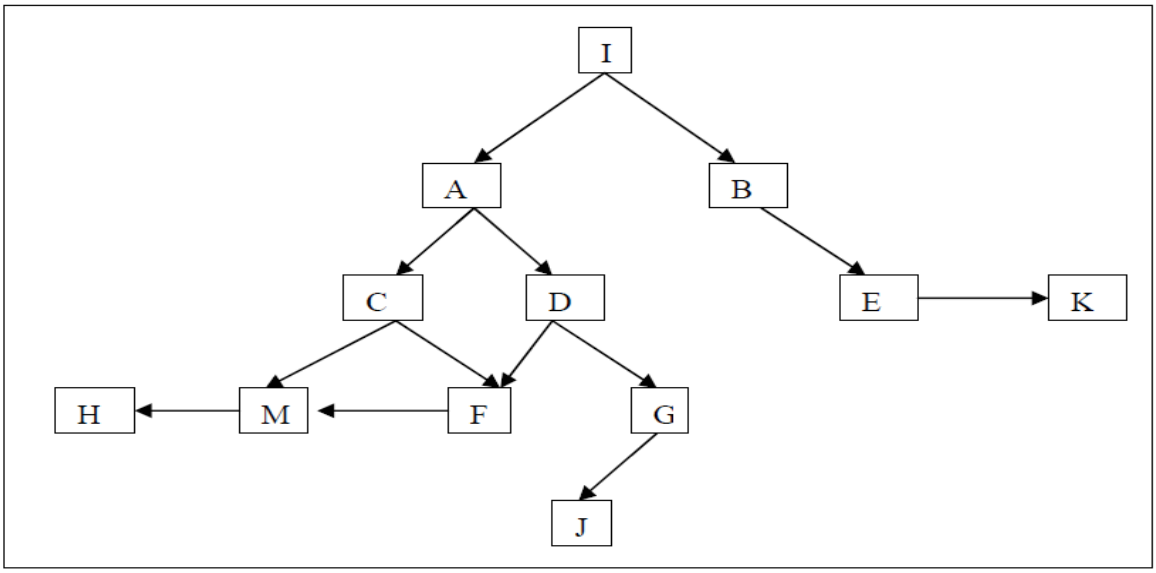

Figure 2.6: Hypertext Structure [20]

2.2.1.5 Browsing Technique

Browsing is an example of a retrieval technique [22] that requires a well structured document collection. Documents must be represented in the system as a network of inter connected nodes and a hypertext is a good way of defining such a structure. With the aid

of the system, the user can now browse through this network to find the information he is looking for. The user does not have to formulate a query at first, to represent his information need and there is no such thing as a formal matching process. This can be a major advantage, especially when the user does not exactly know what he is looking for. In fact, the actual matching takes place in the uses mind while wandering through the network of nodes and links, he decides whether or not he finds the current node interesting. If so, he might want to retrieve the document belonging to the node and we could speak about a match. There are no formal rules for this kind of matching. The user may be changing his mind and decide otherwise or maybe he never really made up his mind about what he is looking for and is not able at all to find any matches [22].

As an example, suppose a user using some kind of drawing program wants to draw a star. He does not know how to do this or even if it is possible at all with the program. So he takes the manual and starts to browse through it to try and find an explanation on how to draw a star. In this case, the nodes could be the names of chapters or sections in the manual, while the links could be the cross-references between these chapters or sections. As a starting point, the user might look for the entry ‘star’ in the index. Reading the particular page referred to in the index, he finds out that it is only a general introduction. However, he finds a reference to a more specific section on the subject. In this way the user will continue to follow cross-references until he finds some description on how to draw a star, or until he finds out that it is not possible to draw stars with the program. For very large collections of documents, browsing can be rather tedious and inefficient and in such cases querying might be a better idea. We could try to get the best of both and combine them.

2.2.2 Classification Schemes

Daniel Lucredio presents a survey about the main research on component search and retrieval [23] in 2004 and discusses how efficiently components can be search in order to support future component markets. They specially focus on the classification schemes used to store the software components. The author proposes the utilization of a facet based scheme to classify the software components and the limited numbers of

characteristics i.e. facets are retrieved with a set of possible keywords are associated with each facet but problem arises when 2 different people may choose different keywords to describe the components. So, to overcome the problem, maarek [24] purposes the use of automatic indexing and extraction of terms or phrases that describe a best component.
There are two main classification schemes- Faceted classification and Enumerative classification scheme.

2.2.2.1 Faceted Classification Scheme

Faceted classification avoids the enumeration of component definitions in a hierarchy by defining attribute classes that can be instantiated with different terms [26]. Terms are grouped into a fixed number of mutually exclusive facets. In this, users search for components by specifying a term for each of the facets. Within each facet, classification techniques are used to help users choose appropriate terms. Basically, classification in a faceted scheme is straightforward. A faceted scheme may have several facets and each facet may have several terms. A hypothetical faceted scheme has two facets, entities and activities, is introduced here to illustrate the classification process.

{entities} {activities}
Systems programming

To classify the title in our example, we select from each facet the term that best describes each of the concepts in the title. "Systems" is selected from the entities facet, and "programming," from the activities facet.

Problems with faceted classification:

- Knowledge of domain expert is implicit
- Complicates the automation
- Automation of classification requires reverse engineering from source code.(source code analysis)

2.2.2.2 Enumerative classification scheme

Classification schemes such as Library of Congress and Dewey Decimal are Enumerative. In an Enumerative scheme all possible classes are pre- defined. The librarian selects the class that best fits the attributes of the new item by traversing the classification hierarchy. In (entities) the Dewey Decimal System, for designs example, if we want to classify the programs title "Structured Systems Program- structures ruing," we look at the hierarchical systems structure and try to find the predefined class that best describes the title but they add a complex network of relationships that is difficult to represent and maintain.

Problems with enumerative classification:

- There is inherent inflexibility present in this scheme.
- It requires users to understand the structure and contents of repository to effectively retrieve the information.

2.2.3 Existing approaches for effective search and retrieval process

There are various existing approaches for the effective search and the retrieval of the components.

2.2.3.1 Four Layered Architecture Approach

Wang Haitao and Chen Xing purposes a component library model based on the four-layered architecture in 2011 [27] for organizing and managing the software component library more efficiently so that user can easily retrieve, understand and reuse the component. The four layered architecture include framework layer, component layer, leaf class layer and leaf function layer. This architecture is particular suitably for the flexibility and reusability of the component library system. This four layered architecture approach basically organize and manage a large number of reusable components effectively and provide tools to support the users retrieve, understand and reuse components conveniently in a successful software development process.

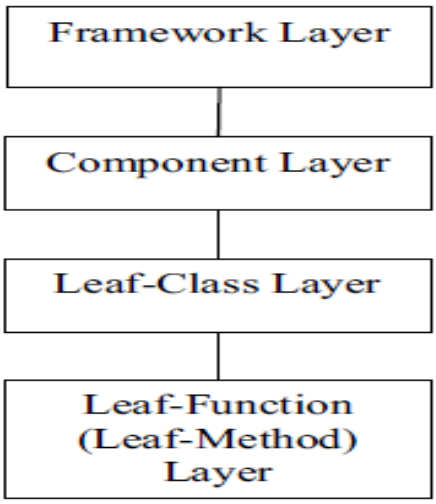

Figure 2.7: Four Layer Architecture of the component library [27]

A component library is divided into four layers, including the framework layer, component layer, leaf class layer and leaf function (leaf method) layer.

- **Framework Layer:** The software framework is the software construction framework determined by the software architecture in a specific domain. Generally, it is a reusable component. The difference compared to ordinary components is that it is not used to solve a local problem but to describe the overall structure of software, including main components and the relationship among them without the details of each component. The software framework studies quick and reliable ways to construct the system from reusable components with a focus on the overall structure of the software [27].
- **Component Layer:** The concept of reusable components is no longer limited to source code components. It is a very broad concept, covering all reusable products in all stages of software lifecycle, including project planning, requirement definition, analysis model, design model, document, tests, data and other useful information during development activities. Generally, any part of the system that can be clearly identified and can be reused by others is called a component [27].
- **Leaf Class Layer:** The class in the leaf class layer only provides services, does not reuse the connective class. It is also characterized by its small scale and a common interface and function. A class defined as such is easy to reuse, and we

refer to it as the leaf class. There are one or more classes in a leaf class. When a leaf class consists of multiple closely connected classes, it only provides the service interface, users only need to know the external interface, not the details inside the leaf class. Such a structure appears externally as an enclosed unit without any connection with external components. As a result, it has a high degree of reusability, and is widely used in other components and systems [27].

- **Leaf Function layer:** The function (method) in this layer only provides services and does not reuse the connective function (method). Its features are small scale, and a common interface and function. This layer is called the leaf function (leaf method) layer, which is the lowest layer of the architecture. It can only be reused by other systems or components. As the leaf node of the component tree, it cannot reuse other components.

2.2.3.2 Knowledge Based Approach

The text automatic classification method is based on the content analysis automatically to allocate the text into pre-determined catalogue. The methods of text automatic classification mainly use information retrieval techniques. Traditional information retrieval mainly retrieves relevant documents by using keyword-based or statistic-based techniques [28]. Knowledge-based software retrieval systems make some kind of lexical, syntactic and semantics analysis of natural language specifications of software components without pretending to completely understand the document. They are based on a knowledge base which stores semantics information about the application domain and about natural language itself. These systems are usually more powerful than traditional keyword retrieval systems. However, they usually require enormous human resources: Knowledge bases are created for each application domain and are usually populated manually.

2.2.3.3 Automatic Indexing Based Approach

Automatic indexing is the process of analyzing an item to extract the information to be permanently kept in an index. The system extracts lexical, syntactic and semantic information from the natural language description. These approaches automatically

extract words or phrases (usually pairs of terms) from documents and from queries in natural language to build their internal representation. Automatic indexing is required for the effective retrieval of information. Indexing is essential to information retrieval because it provides entry points to a collection, without the user having to examine the whole collection. It tells the user where the information can be physically found and allows them to search a collection using keywords or phrases. Indexing has existed for as long as humans have been keeping written records.

There are two ways that information can be indexed; manually or automatically. In manual indexing a human indexer compiles the index, while automatic is when the task is done by computer. Systems that provide automatic indexing of software components can be classified in two basic groups: systems that work only at the lexical level and systems that include Syntactic and Semantic analysis of software descriptions [29].

2.2.3.4 Automatic Tags Extraction (ATE) Based Approach

Lei Zhang, LichaonChen, Lihu Pan, Yingjun Zhang proposed the novel approach Automatic Tags Extraction (ATE) algorithm in 2012 [30] designed for the component retrieval for improving the performance and effectiveness of large scale repositories. In this approach, component tags are extracted automatically and indexed by ATE algorithm then they use the vector space mode (VSM) similarity algorithm to match the component tags. The application of ATE algorithm in description documents of components will find more useful information and significantly improve the recall ratio; the precision ratio is also raised by applying the similarity algorithm. Since applying index mechanism, the ATE retrieval can achieve components retrieval with a fast speed. The process of components retrieval is shown in Figure below.

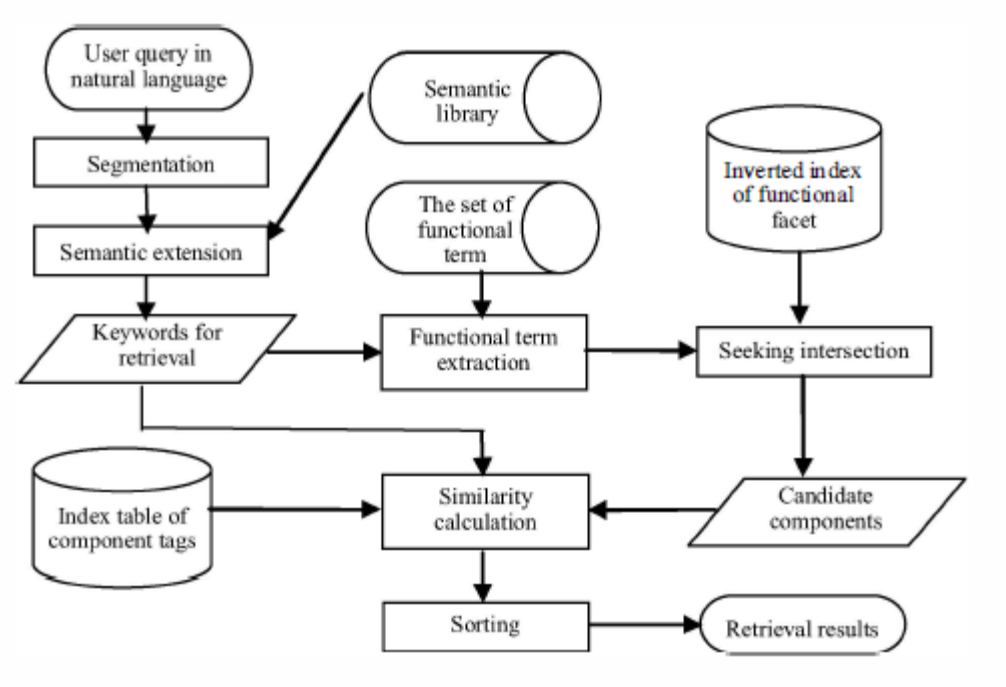

Figure 2.8: ATE Retrieval process [30]

2.3 Effective search and retrieval methods

There are various methods for searching and the retrieval of the components for the effective retrieval. The various methods are discussed as below:

2.3.1 Text Based Search Method

In this case, the entire surrogate is searched and the components found valid according the given search are displayed.

2.3.2 Keyword Based Search Method

In this case the user will only input the keywords which according to him are relevant and shall provide with the required components, once the keywords are filled in, the system checks for only the keywords present in the surrogate and once the relevant keywords are found, the system shall display the related components.

2.3.3 Signature Based Search Method

The user can also base his search on the basis of any particular signature of the components. This type of search helps to provide with components with approximately the exact match from the requirement.

2.3.4 Operational Semantic Based Search Method

In operational semantic keyword based search, the user will give as input i.e. component name and its prototype, then the relevant component is displayed along with its input and output. In this we match the candidate asset against a user query on the basis of the candidate behavior on sample inputs. Here the search is based on the input and output. The operational semantic based search method provides the exact retrieval of the component. Hence precision and recall ratio is better in this case as compare to the above methods.

2.4 Problems faced in traditional search methods

Consider a large university library. Tens of thousands of books, periodicals, and other information resources are available for use. But to access these resources, a categorization scheme must be developed. To navigate this large volume of information, librarians have defined a classification scheme that includes a library of congress classification code, keywords, author names and other index entries. All enable the user to find the needed resource quickly and easily. Similarly in component library, tens of thousands components are stored. How users can finds out the appropriate one. The solution to this problem is classifying the software libraries in a particular order and use effective retrieval technique.

Another Problem rises with the existence of these software assets or components are the presence of a storage structure or a repository that would contain these components and also provide for an easy search and retrieval of the same. The repository has to be constructed and maintained in order to maximize the use of reusable components. In addition to construction of the repository, some retrieval technique also needs to be implemented that utilizes minimal repository structure to effectively support the process of finding software components. Once the repository has been constructed then the

information that needs to be stored in it should also be structured in a way such that it should be digitalized, normalized and archived.

After the assets have been properly stored in the repository, the next problem that arises is finding of the appropriate asset by the user and proper query formation for the same purpose. This has to be tackled by applying proper classification schemes which also form an important part of the retrieval process. The classification is important in order to tackle the search problem that arises due to difference in the users thought process and the final query formation as they help in organizing the assets in the repository in an organized manner.

The retrieval of any component is measured according to abstract performance measures such as precision and recall. Hence, the process that needs to be followed for retrieval should guarantee high performance measures. A number of retrieval techniques are in place and the most appropriate one can be chosen according to the requirements and the available resources. Hence, it reduces the effort, time and other resources that go into building the components again and again.

Chapter 3

Proposed Software Repository

During Literature Review, following research gaps have been observed.

3.1 Research Gaps

While doing the Literature survey, we analyze the various research gaps which are as follows:

I. Most of the repository extracts source code only.

II. Repository had problems whenever vocabulary mismatch means problem is in Keyword based retrieval.

III. There is a need to improve the similarity matching algorithm to further increase the precision ratio.

IV. There is a need to improve the effectiveness of the component retrieval.

V. Installation problems need to be manually addressed by the component maintainers.

VI. Retrieval of Multiple related components are not much effective.

VII. Ranking the components should be used based on how close a component is too classified according to the high precision and high recall.

VIII. Operational Semantic interpretation of the user queries is required against the software repository.

3.2 Problem Formulation

Every software component can be reused for some other purpose. The problem that is generally encountered while doing so is the efficient search and retrieval of these

components from the software library as they are not properly represented. So, our basic task is to identify the general challenges, which are encountered while storing and retrieving the components, which have some reuse potential.

In order to do so, the first requirement is a user friendly software repository .So that there is maximum precision and recall of the visited and retrieved software components.

The basic aim is to develop software, which facilitates this and to provide for the maximum relevancy of the retrieved components such that, they can either be reused in the exact same manner or can be modified to do so.

There are many software's and websites which provide for the retrieval of reusable components, but, the search is not precise and the retrieval not that exact. So, the system proposes to make the search as concise and precise as possible and also facilitate the users with various retrieval methods, out of which they can select the one they deem fit for their purpose.

The user should be able to perform search based on certain fixed categories by using different retrieval techniques i.e. Keyword based retrieval, signature based retrieval and operational semantic based retrieval. The basic comparison between these techniques on the basis of precision and recall should be done for improving the above research gaps.

Thus, this thesis deals with the construction of the software repository, Construction of the operational semantic keyword based retrieval technique for evaluating the best results and at the end comparison between the techniques for retrieval of the software components is made on the basis of precision and recall.

3.3 Objectives of the Thesis

Based on the above problem statement, we have set following objectives:

I. To identify the issues related to the search and the retrieval of the components faced by the user.

II. To develop a software repository which extract the source code, documentation and executable files.

III. To implement the operational semantic keyword based retrieval for the precise search and the effective and the exact retrieval of the components.

IV. To compare the operational semantic search technique with the signature based technique and the keyword based technique on the basis of some fixed criteria i.e. precision and recall to find out the efficient retrieval.

3.4 Methodology used in the proposed system

Following methologies are used to implement the above objectives.

I. Various existing retrieval techniques have been studied in literature review. During literature review certain gaps were found and few of them are implemented by the purposed repository.

II. A software repository has been developed to remove some research gaps in vb.net and sql server.

III. Implementation process is as follows

   a) Firstly keyword based and signature based retrieval techniques have been implemented.
   b) Then operational semantic keyword based retrieval technique has been implemented in the purposed repository.
   c) Comparison is done between these retrieval techniques for the precise search and the effective and the exact retrieval of the components on the basis of precision and recall.

IV. Various types of testing on the repository have been done. Test cases are performed on the purposed repository for various values and experimental results have been observed on the basis of precision and recall.

Chapter 4

Design and Implementation Details

This Chapter includes the implementation details for the proposed software repository by using effective retrieval techniques .VB.net and SQL server is used for the implementation. It integrates computation, visualization and programming in easy to use environment. As discussed in Literature survey, there are various techniques, approaches and methods having their own merits and de-merits. In our implementation, we used the advanced keyword based search, signature based search and the operational semantic keyword based search for the effective search and retrieval of the software components.

4.1 Features of proposed Software Repository

The main features of the proposed system are:

I. It contains a dedicated repository for storage of components that can be reused.

II. It contains an interface, which provides with the various criteria of reusable components for the user to confine his search.

III. The interface provides various effective search and retrieval techniques for the exact required components.

IV. The Administrator manage the storage , updation and deletion of components.

V. User can easily download the executables, source code and the documentation of the exact components.

VI. The software library provides the signature based search, in which the user can search the components on basis of parameters and the return types.

VII. This Repository also provides the operational semantics keyword based search, in which the user can search the components on the basis of input output.

VIII. To compare the signature based search and operational semantic keyword based search with the traditional keyword based search on the basis of some fixed criteria i.e. precision, recall and the coverage ratio to find out the effective retrieval.

4.2 UML Specification of the repository: Data Flow Diagram.

The Proposed repository consists of two main modules i.e. Component Search Engine and Component Storage Management which are shown as follow.

4.2.1 Component Search engine

This Module provides the functionality of searching the appropriate component using techniques namely advanced keyword based search, signature based search and operational semantic keyword based search techniques.

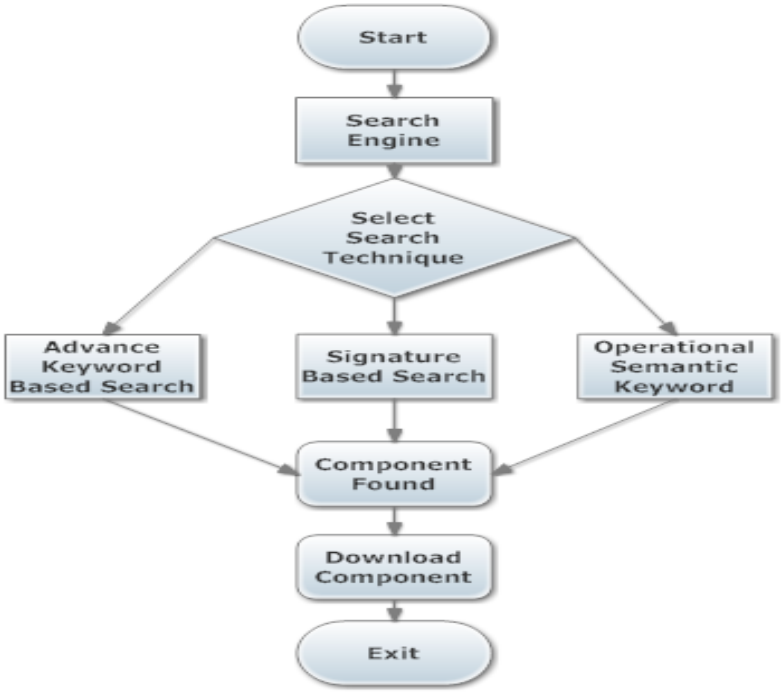

Figure 4.1: Flow of component search module.

4.2.2 Component Storage management

This module allows the administrator to manage the storage, updation and deletion of the components. She can also add operational keywords for the operational semantic based search.

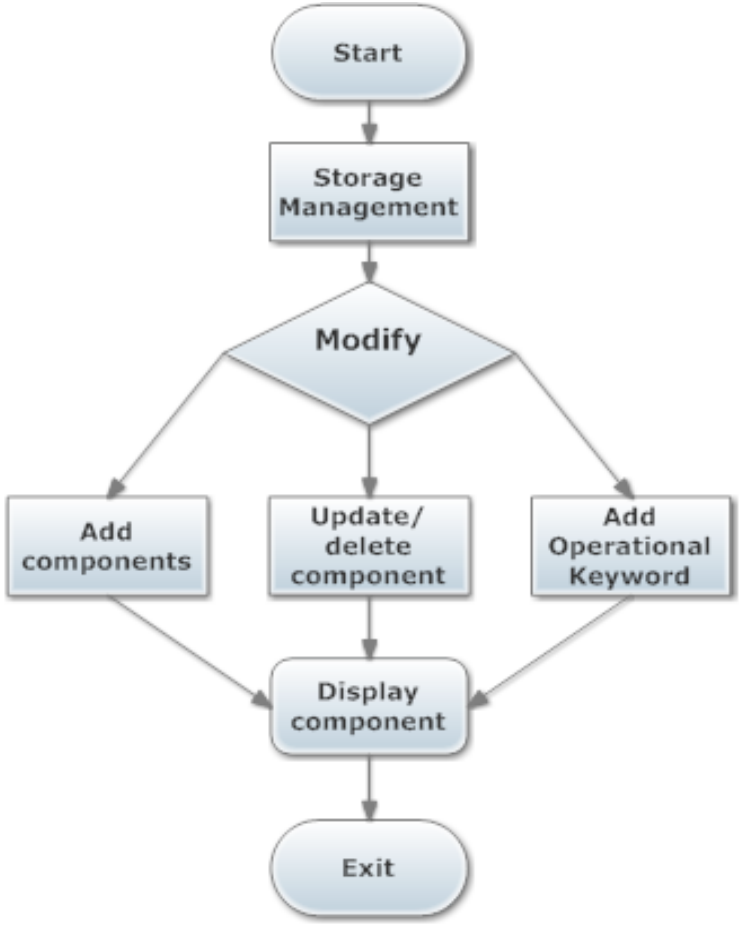

Figure 4.2: Flow of component storage management.

4.3 Working of proposed repository

Software Retrieval is a web application, the working and functionalities of its various units are described in this section.

4.3.1 Admin and User Authentication

The user has to login every time they wish to access the site and the components available. The user can either be a registered user or the administrator. Figure 4.3 shows the User Login area where the user has to enter his authenticated username and password to access the system.

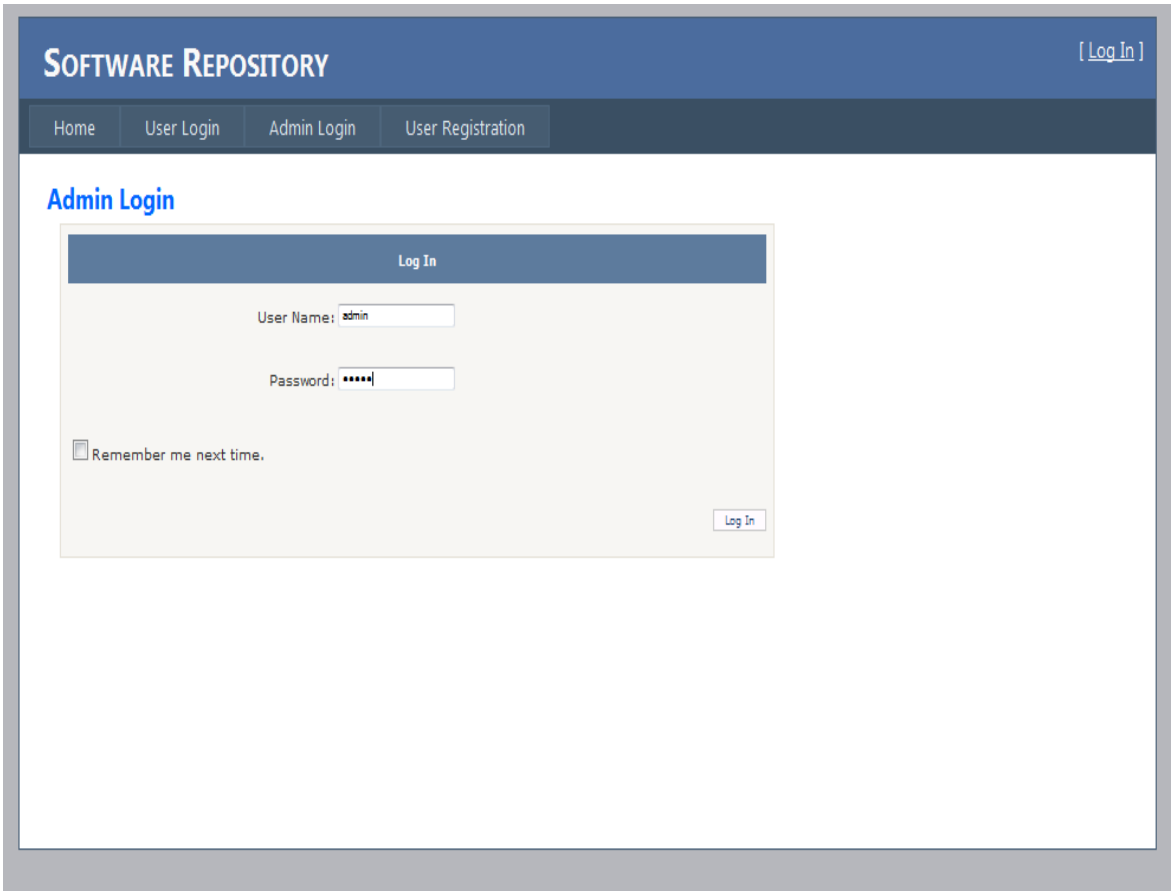

Figure 4.3: Snap Shot of User and Admin Login Area

4.3.2 User Registration

The first step for any user is to register himself so that he may be given the access rights to not only view but also download the available components.

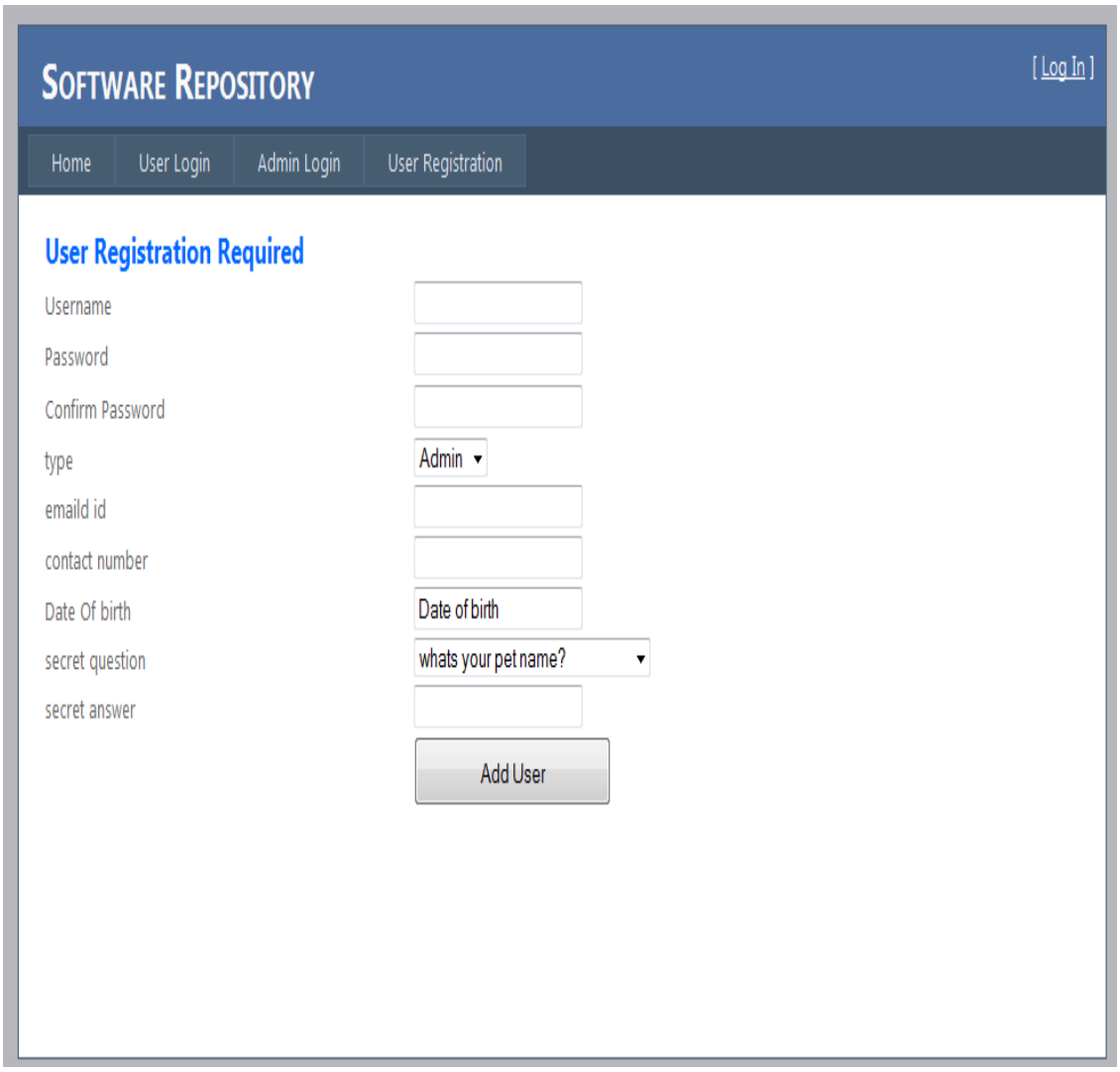

Figure 4.4: Snap Shot of user registration

4.3.3 Component Search Engine

Software repository mainly contains 2 modules i.e. component search engine and component storage management. User can choose component search engine module. After choosing component search engine, user can search the components by using various techniques namely advanced keyword search, signature based search and operational semantic keyword based search.

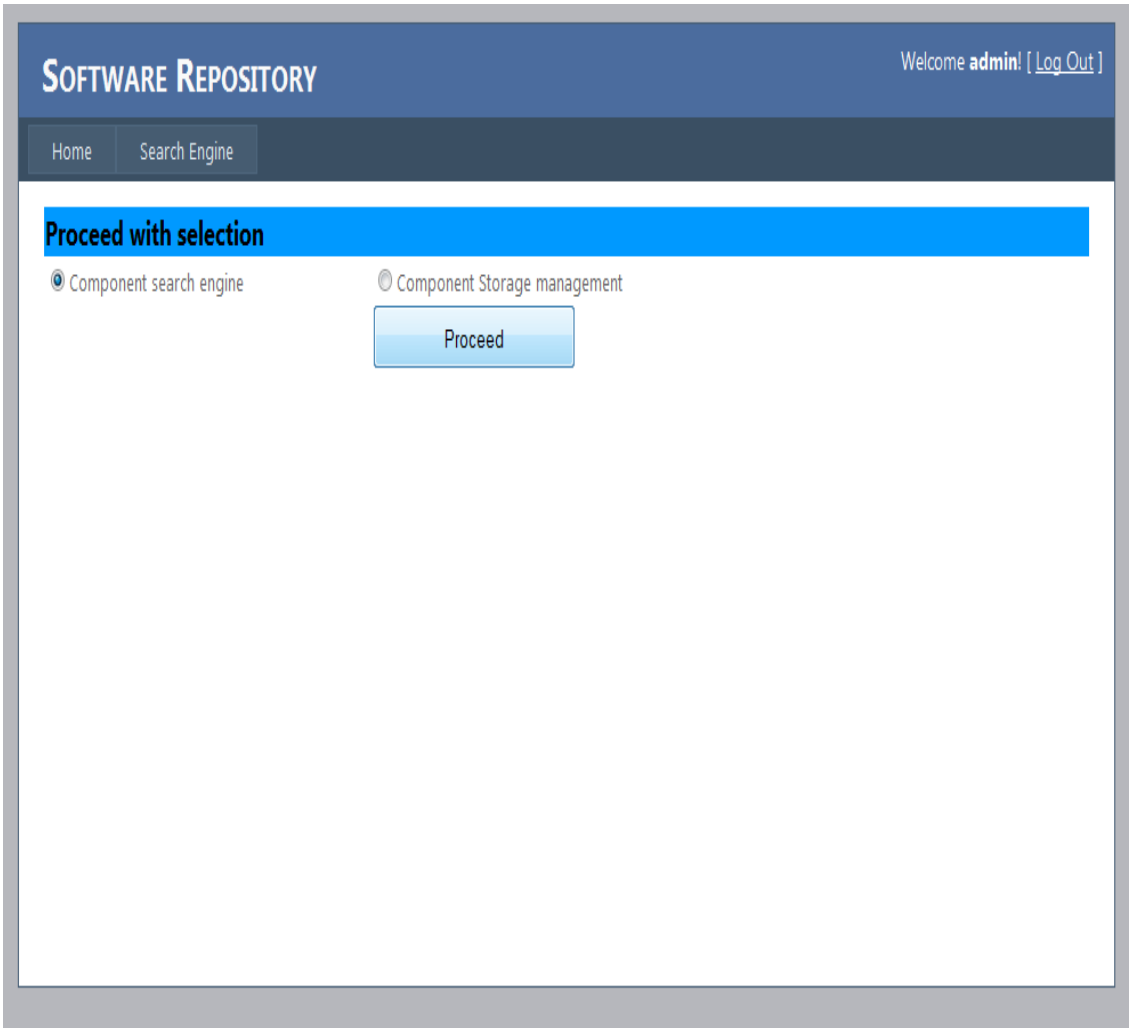

Figure 4.5: Snap Shot of Component search engine

4.3.4 Keyword Based Search

In this case the user will only input the keywords which, according to him are relevant and shall provide with the required components, once the keywords are filled in, the system checks for only the keywords present in the surrogate and once the relevant keywords are found, the system shall display all the related components as in Figure 4.5.

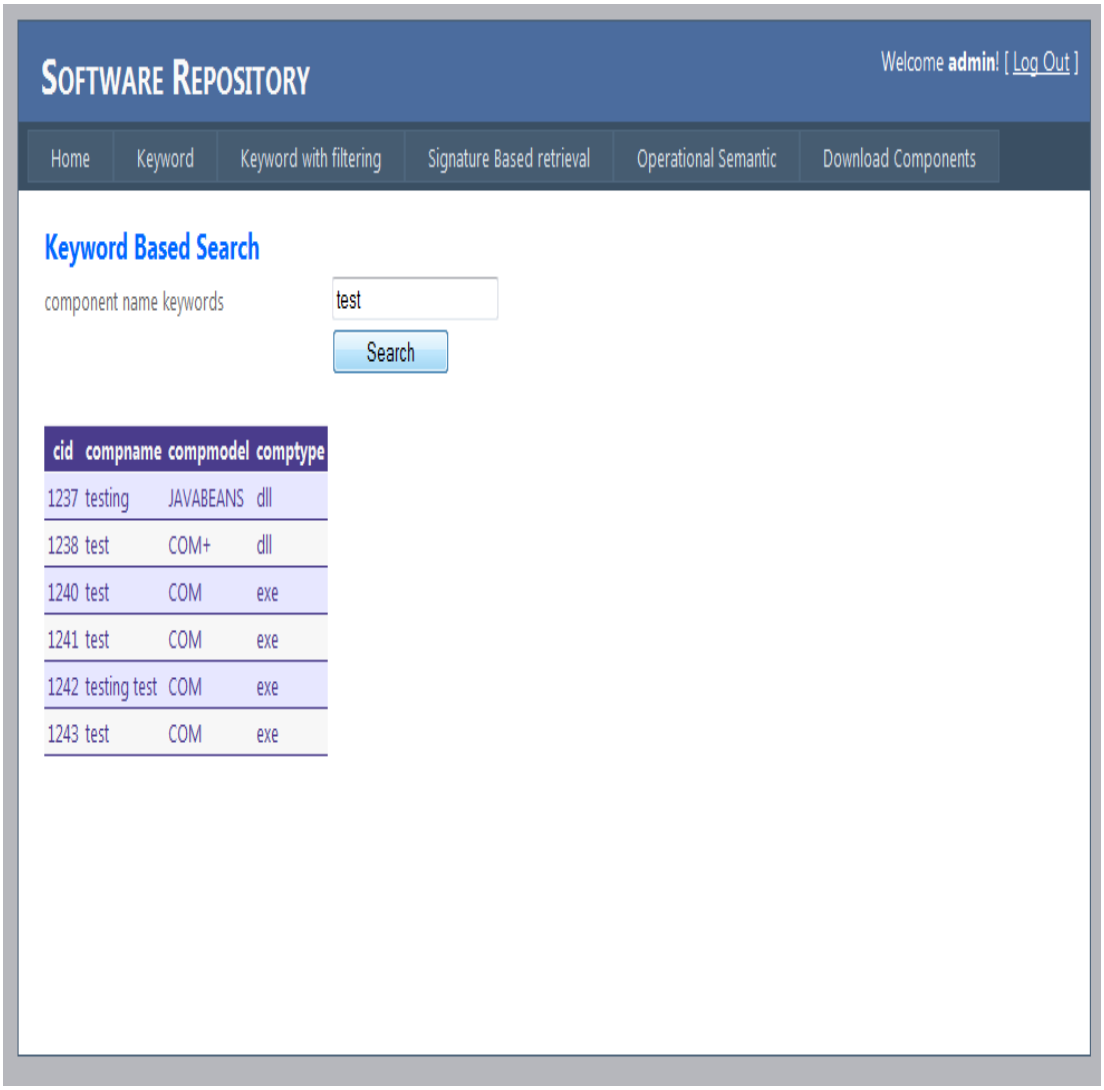

Figure 4.6: Snap Shot of Keyword based search

4.3.5 Keyword Advanced Search

In Advanced Keyword Search, user will search the component according to the type of component and on the basis of component model.

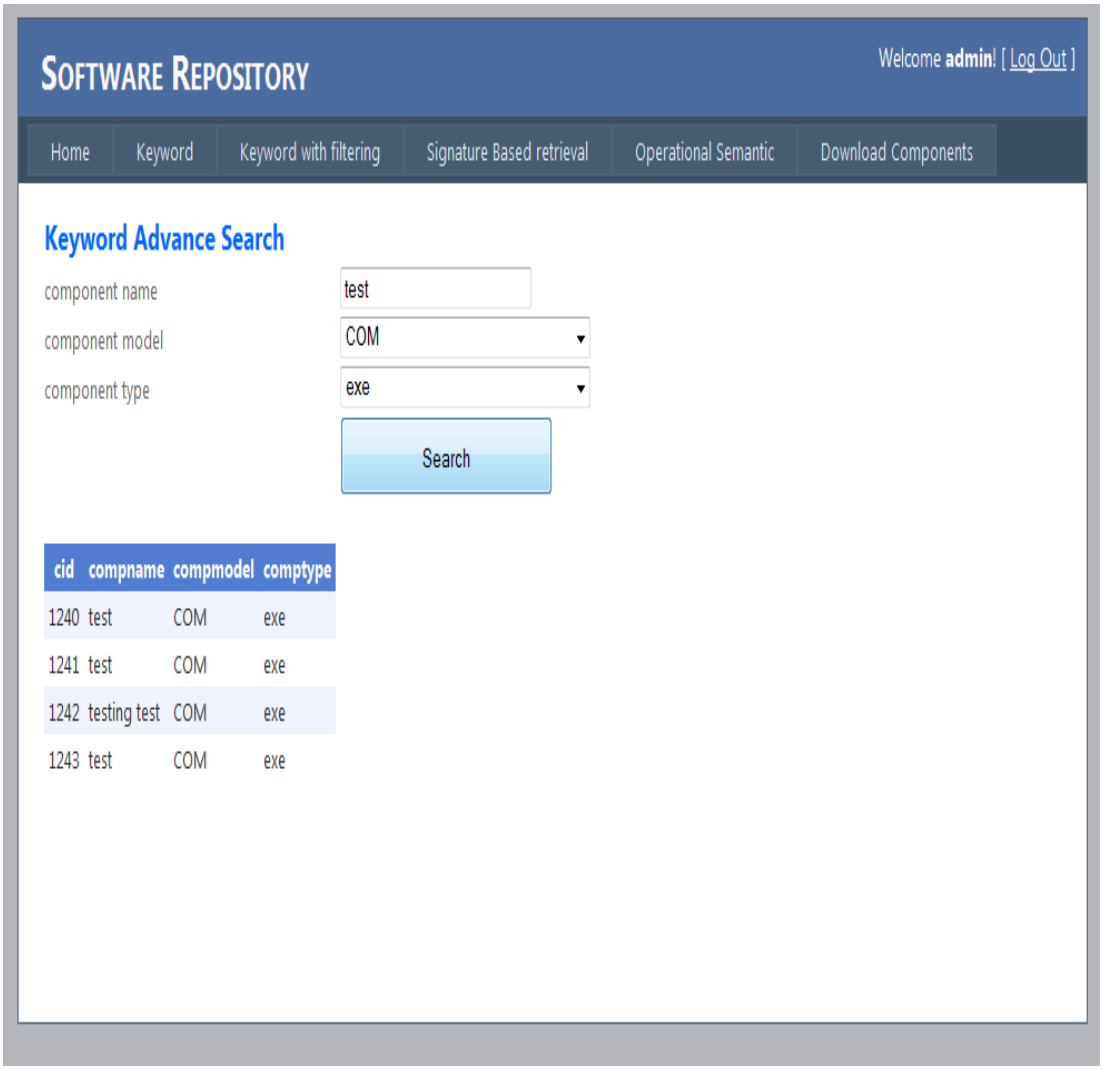

Figure 4.7: Snap Shot of Advanced Keyword based search

4.3.6 Signature Based Retrieval

The user can also base his search on the basis of any particular signature of the components i.e. on the basis of prototype of parameters, number of parameters and return type. This type of search helps to provide with components with approximately the exact match from the requirement.

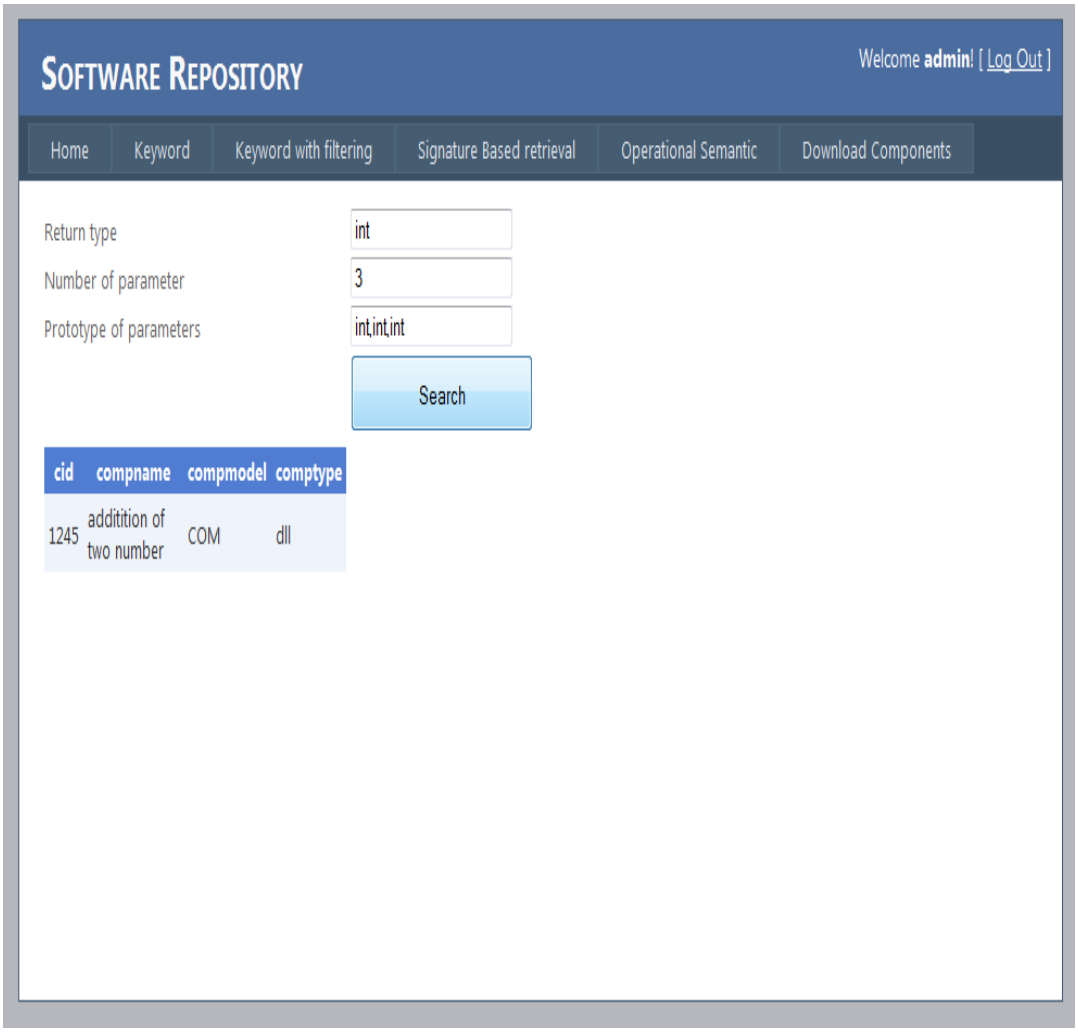

Figure 4.8: Snap Shot of Signature based retrieval

4.3.7 Operational semantic keyword based search

In operational semantic keyword based search, the user will give as input i.e. component name and its prototype, then the relevant components are displayed along with its input and output.

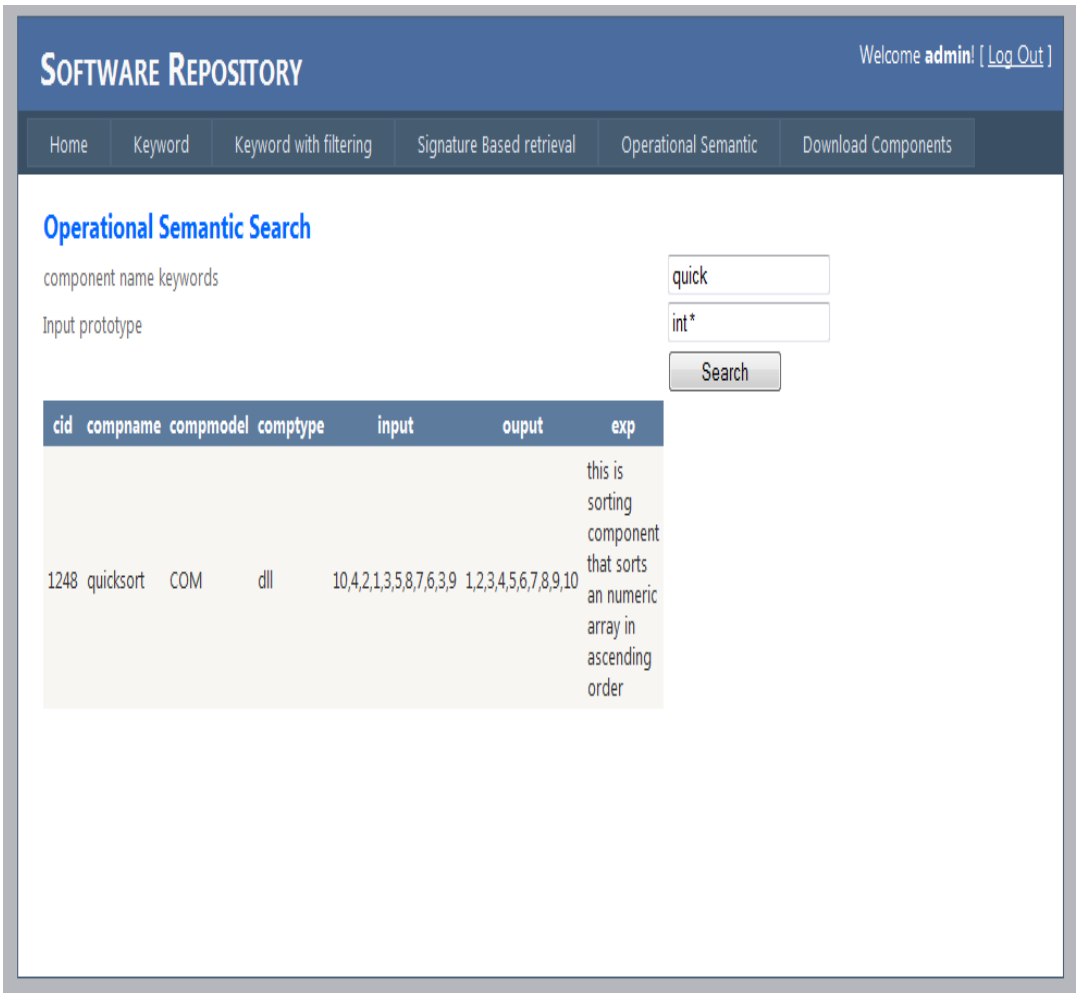

Figure 4.9: Snap shot of Operational semantic keyword based search

4.3.8 Download Components

After searching the relevant component according to the need .User can easily download the source code, executables and documentation of the component.

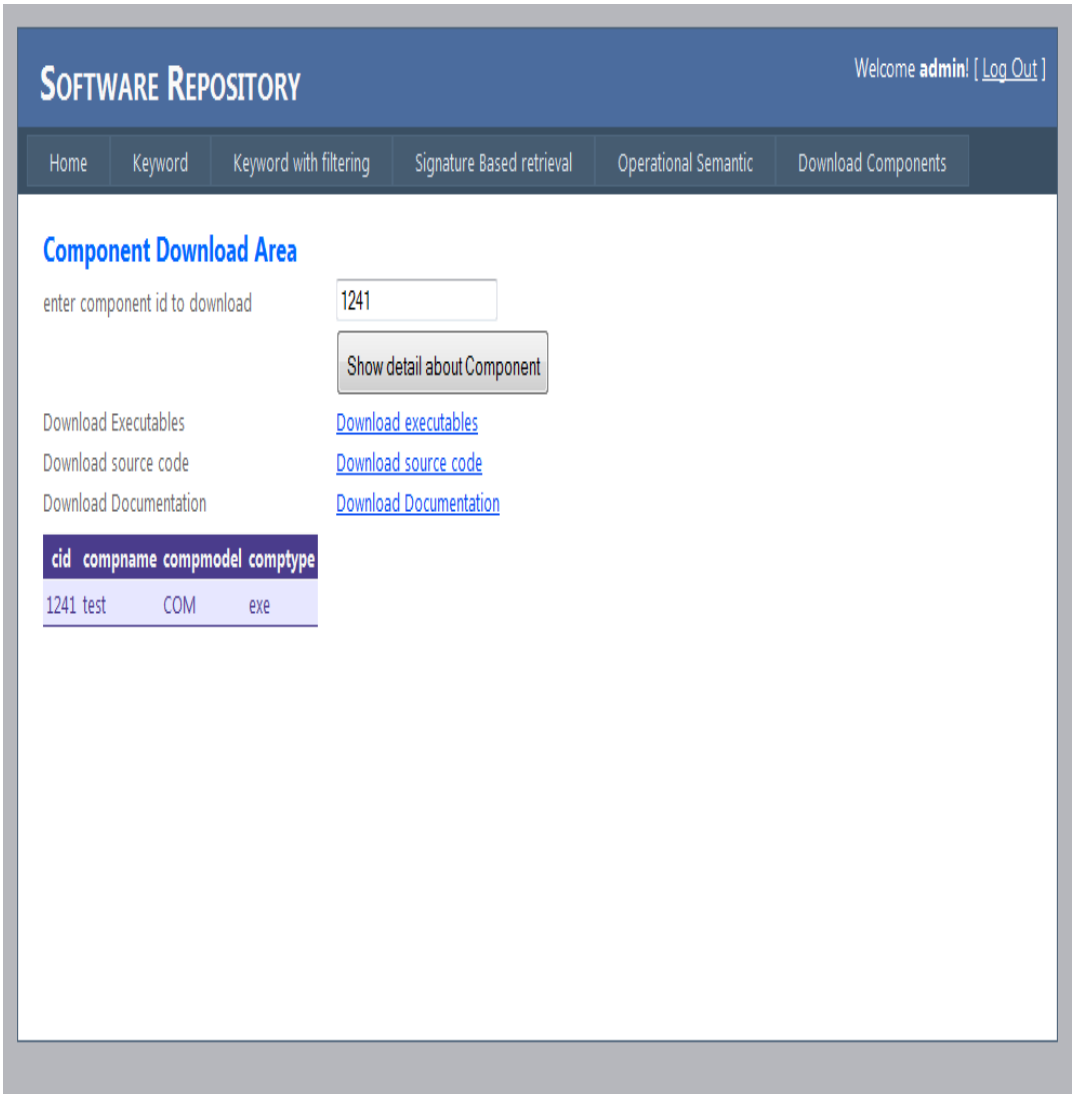

Figure 4.10: Snap shot of download component

4.3.9 Component Storage Management

This module allows the administrator to manage the storage, updation and deletion of the components. She can also add operational keywords for the operational semantic based search.

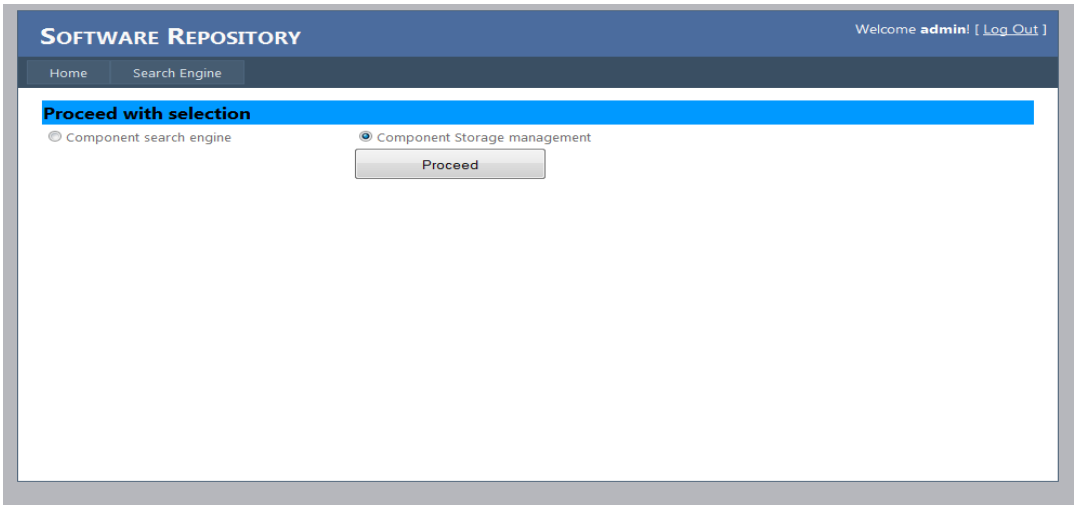

Figure 4.11: Snap shot of component storage management

4.3.10 Add components

Admin can add the components according to the type of the component, component model, description, input, output and upload the respective executable, source code and the documentation.

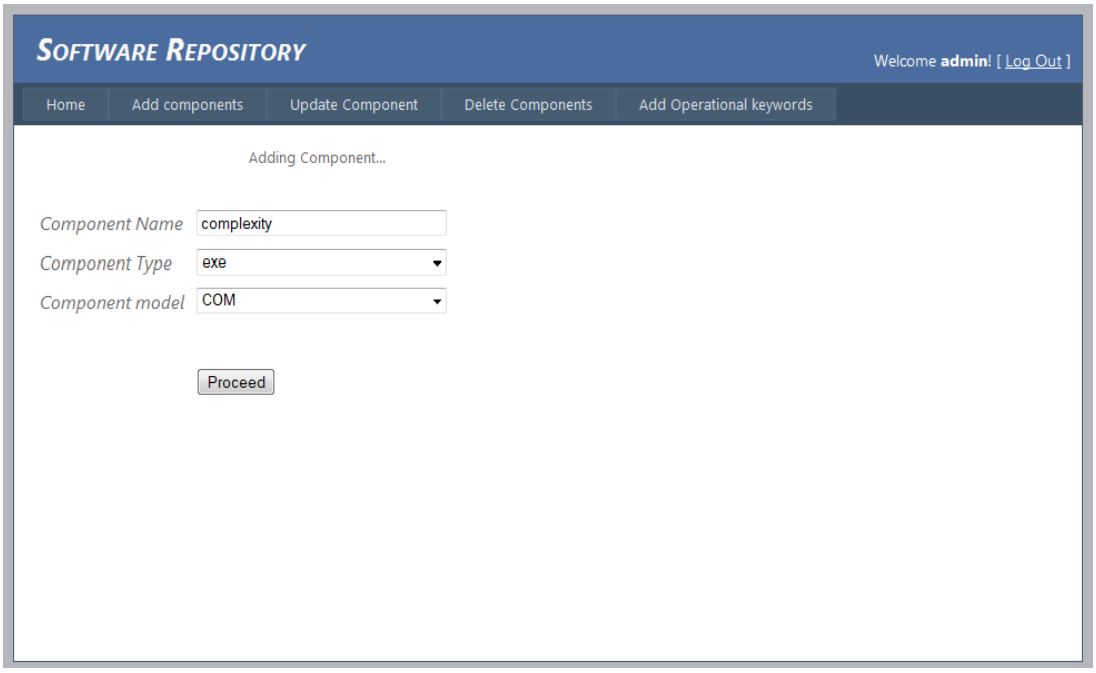

Fig 4.12 Snap shot of add components

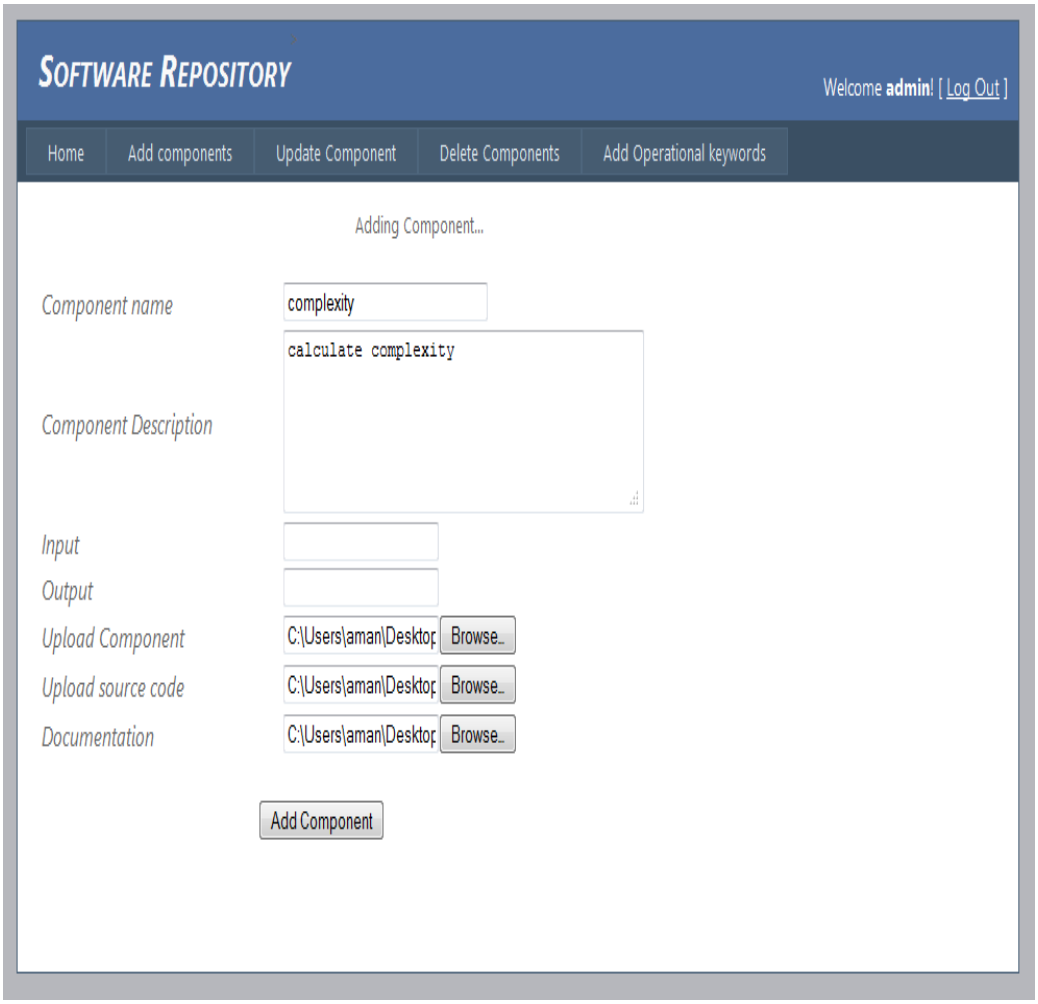

Fig 4.13 Snap shot of add components

4.3.11 Update Components

Admin can update any present components by giving new input output, description or by uploading the new executable, source code and the documentation.

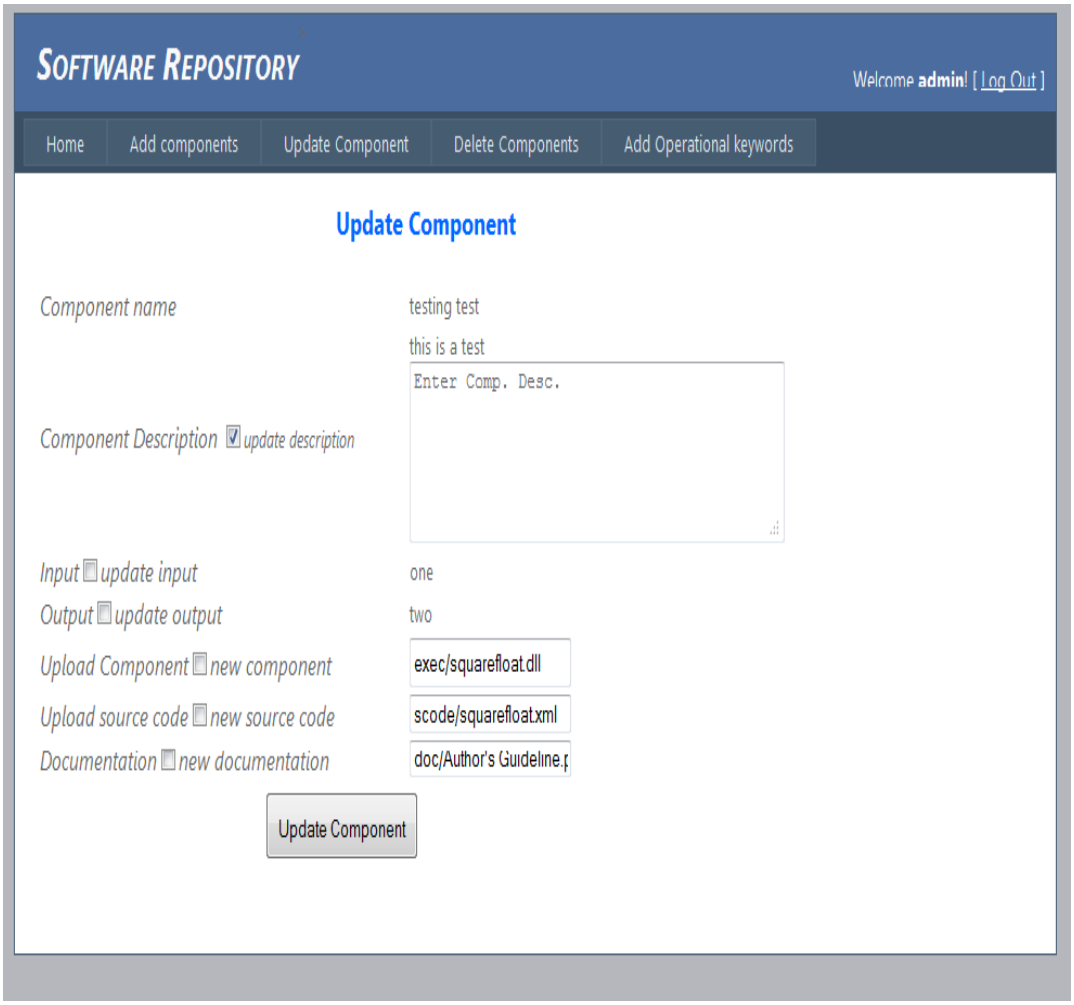

Fig 4.14 Snap shot for update components

4.3.12 Delete Components

Admin can delete any existing component by giving component id and component name. The appropriate component will be deleted from the software repository.

Fig 4.15 Snap shot for delete components

4.3.13 Add operational Keywords

Admin can also add the new operational semantic keyword by giving component id for the operational semantic keyword search.

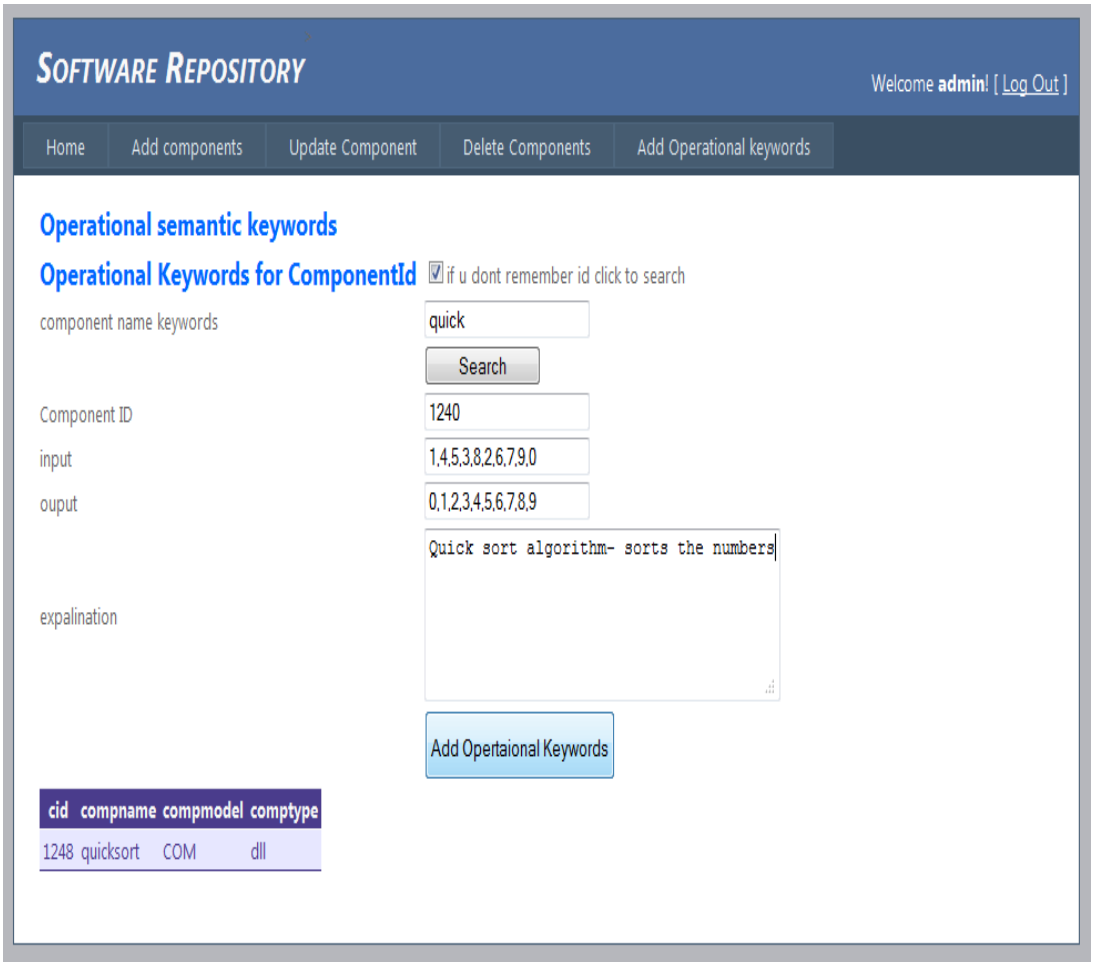

Fig 4.16: Snap shot for adding operational keywords

Chapter 5

Testing and Experimental Results

This chapter includes all the testing details and the experimental results obtained after testing the purposed repository on the basis of precision and recall value.

5. 1 Testing

Software testing can be stated as the process of validating and verifying that a computer program/application/product meets the requirements that guided its design and development, works as expected, can be implemented with the same characteristics and satisfies the needs of stakeholders. Software testing can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted [31].

Testing improves the quality of the software by identifying the requirements that might not be testable and helps in determining the overall product feasibility.

5.2 Types of Testing done on the Proposed Repository

The system has undergone many levels of testing methodologies. Basically this section explains how the system has been tested with them. Following types of testing are performed on this repository.

5.2.1 Unit Testing

Each and every unit of the system has been tested independently. The various units include:

- Administrative unit
- User unit
- Component search engine.
- Component storage management.

The Result of unit testing indicates that the individual units of the tool are working correctly as per the requirement.

5.2.2 Integration Testing

Integration testing has been done by combining individual software modules and tested as a group. It is usually done after the unit testing. The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items.

Thus the entire system after being integrated into one website has also been tested as a whole by using the top down approach to find out the proper working of each and every unit independently as well as a part of the entire system.

5.2.3 System Testing

System testing is the testing of behavior of a complete and fully integrated software product based on the software requirements specification (SRS) document. The main focus of this testing is to evaluate Functional and End-user requirements.

Hence, Each and every objective of the system has been verified in order to find out whether they have been incorporated in the system or not and it has been found that all objectives have been achieved.

5.2.4 Alpha Testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developer's site. Alpha testing is often employed for off-the shelf software as a form of internal acceptance testing, before the software goes to beta testing.

A set of 6 users were taken to test the system and each one was asked to create an account. The functioning of the system was then tested and changes were incorporated in the system.

5.2.5 Regression Testing

Each time a new module is added as a part of integration testing, the software changes. New data flows paths are established, new I/O may occur and new control logic is invoked. These changes may cause problems with functions. Thus it is the activity that helps to ensure that changes (due to any reason) do not introduce unintended behavior or additional errors.

Regression testing has been done on the system after doing changes in the module.

5.3 Test Cases for the proposed software repository

Various test cases are designed module wise during the testing. The details are as below:

5.3.1 Test Case No.1:

Test Objective: - To check validity of the entered username and password.

Test Data: - User Name and Password.

Table 5.1: Login authentication

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter User Name and press LOGIN Button.	User Name correct	Should Display Warning Message "password required"	Display Warning Message "password required"	PASS
2.	Enter Password and Press LOGIN Button.	Password correct	Should Display Warning Message "username required"	Display Warning Message "username required"	PASS
3.	Enter User Name and Password and press LOGIN Button.	User Name correct and Password incorrect	Should Display Warning Message "invalid username and password"	Display Warning Message "invalid username and password"	PASS
4.	Enter User Name and Password and press	User Name incorrect and Password	Should Display Warning Message "invalid username	Display Warning Message "invalid username and	PASS

	LOGIN Button.	correct	and password"	password"	
5.	Enter User Name and Password and press LOGIN Button.	User Name and Password correct acc to user	Should Navigate to keyword.aspx page	Should Navigate to keyword.aspx page	PASS
6.	Enter User Name and Password and press LOGIN Button.	User Name and Password correct acc to admin	Should Navigate to add component.aspx page	Should Navigate toad component.aspx page	PASS

5.3.2 Test Case No.2:

Test Objective: - To check whether the new user is properly registered or not.

Test Data: - The Fields with asterisk (*) sign are compulsory to filled.

Table 5.2: User Registration

S no.	**Steps to Execute**	**Test data/input**	**Expected Output**	**Actual Output**	**Result**
1.	Enter all (*) fields and press 'Submit" button.	Enter all fields except "email_id" and press "Submit" button	Should Display Warning Message "(*) field cannot be leave empty"	Display Warning Message "(*) field cannot be leave empty"	PASS
2.	Enter all (*) fields and press "Submit" button.	Enter all fields except "Date of birth" and press "Submit" button	Should Display Warning Message "(*)field cannot be leave empty"	Display Warning Message "(*) field cannot be leave empty"	PASS
3.	Enter all (*) fields and press "Submit" button.	Enter all fields except "Username" and press "Submit" button	Should Display Warning Message "(*)field cannot be leave empty"	Display Warning Message "(*) field cannot be leave empty"	PASS
4.	Enter all (*) fields and press "Submit" button.	Enter all fields except "Password" and press "Submit" button	Should Display Warning Message "(*)field cannot be leave empty"	Display Warning Message "(*) field cannot be leave empty"	PASS

5.	Enter all (*) fields and press "Submit" button.	Enter all fields except "confirm password" and press "Submit".	Should Display Warning Message "(*)field cannot be leave empty"	Display Warning Message "(*) field cannot be leave empty"	PASS
6.	Enter all (*) fields and press "Submit" button.	Enter all (*)fields and press "Submit" button	Should Display Message " You Are Registered successfully"	Display Message " You Are Registered successfully"	PASS

5.3.3 Test Case No.3:

Test Objective: - To check whether the components download properly or not.

Test Data: - To check "downloading of a component".

Table 5.3: Downloading of the components

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter component id then Press "show details" button. Select the download option.	If user entered not any details.	Message should be Displayed " enter component id"	Message should be Displayed " enter component id"	PASS
2.	Enter component id then Press "show details" button. Select the download option.	If user select the option of "download exe"	Executable will be download.	Executable will be download.	PASS
3.	Enter component id then Press "show details" button. Select the download option.	If user select the option of "download documentatio n"	Documentation download	Documentation download	PASS

4.	Enter component id then Press "show details" button. Select the download option.	If user select the option of "download source code"	Source code download	Source code download	PASS

5.3.4 Test Case No.4:

Test Objective: -To check whether the Keyword based search execute properly or not.

Test Data: Enter the keyword for the search.

Table 5.4: Keyword based search

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter keyword name, model and type then press "submit button".	If user entered not any details.	Message should be Displayed " enter component name"	Message should be Displayed " enter component name"	PASS
2.	Enter keyword name, model and type then press "submit button".	If component name= test and rest fields are empty.	Message should be Displayed " choose component type"	Message should be Displayed " choose component type"	PASS
3.	Enter keyword name, model and type then press "submit button".	If component name= test, type= dll and rest fields are empty.	Message should be Displayed " choose component model"	Message should be Displayed " choose component model"	PASS

5.3.5 Test Case No.5:

Test Objective: -To check whether the signature based search execute properly or not.

Test Data: Enter the return type, number of parameter and parameter prototype.

Table 5.5: Signature based search

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter return type, number of parameters and prototype and then press “submit button”.	If user entered not any details.	Message should be Displayed “ enter return type”	Message should be Displayed “ enter return type”	PASS
2.	Enter return type, number of parameters and prototype and then press “submit button”.	If return type= int and rest fields are empty.	Message should be Displayed “select prototype”	Message should be Displayed “select prototype”	PASS
3.	Enter return type, number of parameters and prototype and then press “submit button”.	If return type= int,number of parameters= 2 and rest fields are empty.	Message should be Displayed “select prototype”	Message should be Displayed “select prototype”	PASS
4.	Enter return type, number of parameters and prototype and then press “submit	If return type= int,number of parameters= 2 and	Relevant components found	Relevant components found	PASS

	button".	prototype= int, int . Then press "search button"			

5.3.6 Test Case No.6:

Test Objective: -To check whether the operational semantic based search execute properly or not.

Test Data: Enter the component name and prototype for search.

Table 5.6: Operational semantic based search

S no.	**Steps to Execute**	**Test data/input**	**Expected Output**	**Actual Output**	**Result**
1.	Enter component name and prototype and then press "submit button".	If user entered not any details.	Message should be Displayed " enter keyword for search"	Message should be Displayed " enter keyword for search"	PASS
2.	Enter component name and prototype and then press "submit button".	If component name= quick sort and rest fields are empty.	Message should be Displayed "mention prototype"	Message should be Displayed "mention prototype"	PASS

3.	Enter component name and prototype and then press “submit button”.	If component name= quick sort and prototype= int*	Relevant components found	Relevant components found	PASS

5.3.7 Test Case No.7:

Test Objective: -To check the addition of components into the software repository

Test Data: Enter the component name, type, model, input, output and description.

Table 5.7: Adding components

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter component name, prototype, model, input, output and description and then press “submit button”.	If user entered not any details.	Message should be Displayed “ enter name”	Message should be Displayed “ enter name ”	PASS
2.	Enter component name, prototype, model, input, output and description and then press “submit	If component name= bubble sort , type =dll, model= com+, input= integers and rest fields are	Message should be Displayed “mention output type”	Message should be Displayed “mention output type”	PASS

	button".	empty.			
3.	Enter component name, prototype, model, input, output and description and then press "submit button".	If component name= bubble sort , type =dll, model= com+, input= integers and output= integers in ascending order and then press "add component".	Component added successfully.	Component added successfully.	PASS

5.3.8 Test Case No.8:

Test Objective: -To check the updating of component in the software repository.

Test Data: Enter the component id, name, type, model, input, output and description.

Table 5.8: Component updating

S no.	**Steps to Execute**	**Test data/input**	**Expected Output**	**Actual Output**	**Result**
1.	Enter component name, id, prototype, model, input, output and description and then press "update button".	If user entered not any details.	Message should be Displayed " choose id first"	Message should be Displayed " choose id first"	PASS

2.	Enter component name, id, prototype, model, input, output and description and then press “update button”.	If component id=1242, name= deleting, type =dll, model= com+, input= integers and rest fields are empty.	Message should be Displayed “mention all details”	Message should be Displayed “mention all details”	PASS
3.	Enter component name, id, prototype, model, input, output and description and then press “update button”.	If component id=1242, name= deleting , type =dll, model= com+, input= integers and all fields are filled.	Component updated successfully.	Component update successfully.	PASS

5.3.9 Test Case No.9:

Test Objective: -To check the deletion of component from the software repository.

Test Data: Enter the component id and name.

Table 5.9: Component deletion

S no.	**Steps to Execute**	**Test data/input**	**Expected Output**	**Actual Output**	**Result**
1.	Enter component id and name and then press “delete button”.	If user entered not any details.	Message should be Displayed “ choose id first”	Message should be Displayed “ choose id first”	PASS
2.	Enter	If user	Message should	Message should be	PASS

	component id and name and then press “delete button”.	entered id=1242 but did not select the component name	be Displayed “ select component for deletion”	Displayed “ select component for deletion”	
3.	Enter component id and name and then press “delete button”.	If user entered id=1242 and select the component and then press the “delete button”	Component deleted successfully.	Component deleted successfully.	PASS

5.3.10 Test Case No.10:

Test Objective: -To add the operational keywords

Test Data: Enter the component id, input, output and description.

Table 5.10: Adding operational keywords

S no.	Steps to Execute	Test data/input	Expected Output	Actual Output	Result
1.	Enter component id, input, output and description and then press “Add keyword” button.	If user entered not any details.	Message should be Displayed “ choose id first”	Message should be Displayed “ choose id first”	PASS
2.	Enter component id, input, output and	If user entered id=1242 but did not select	Message should be Displayed “mention input and output”	Message should be Displayed “mention input and output”	PASS

	description and then press "Add keyword" button.	the component name			
3.	Enter component id, input, output and description and then press "Add keyword" button.	If user entered id=1242 and all the other details are mentioned	Operational keyword added successfully.	Operational keyword added successfully.	PASS

5.4 Results Obtained After Testing the System

Some results on the basis of the two main parameters of precision and recall ratio have been derived in order to find out the best retrieval technique out of the ones implemented. Also the comparison has been made between as follows.

5.4.1 Result obtained by Precision and Recall ratio of Keyword Based Retrieval.

Suppose there are many components which are stored in the software repository. For example we want to search the component of "addition" i.e. addition of two numbers or addition of two strings. Suppose we search this component from the software repository and compared the result on the basis of precision and recall. The three components have been randomly selected from the software repository which is shown in below table 5.11.

Table: 5.11 Randomly Selected Components

Component ID	Component Name	Component Model	Component Type
1254	Addition	Corba	dll
1253	Addition	Java bean	Source code
1257	Addition	Com+	dll

The following result is obtained by the Keyword-based Search on the basis of precision and recall as shown in below table 5.12.

Table 5.12: Result obtained for Keyword Based search

Comp ID / Comp Criteria	**1254**	**1253**	**1257**
Precision	0.25	0.125	0.25
Recall	0.16	0.375	0.166

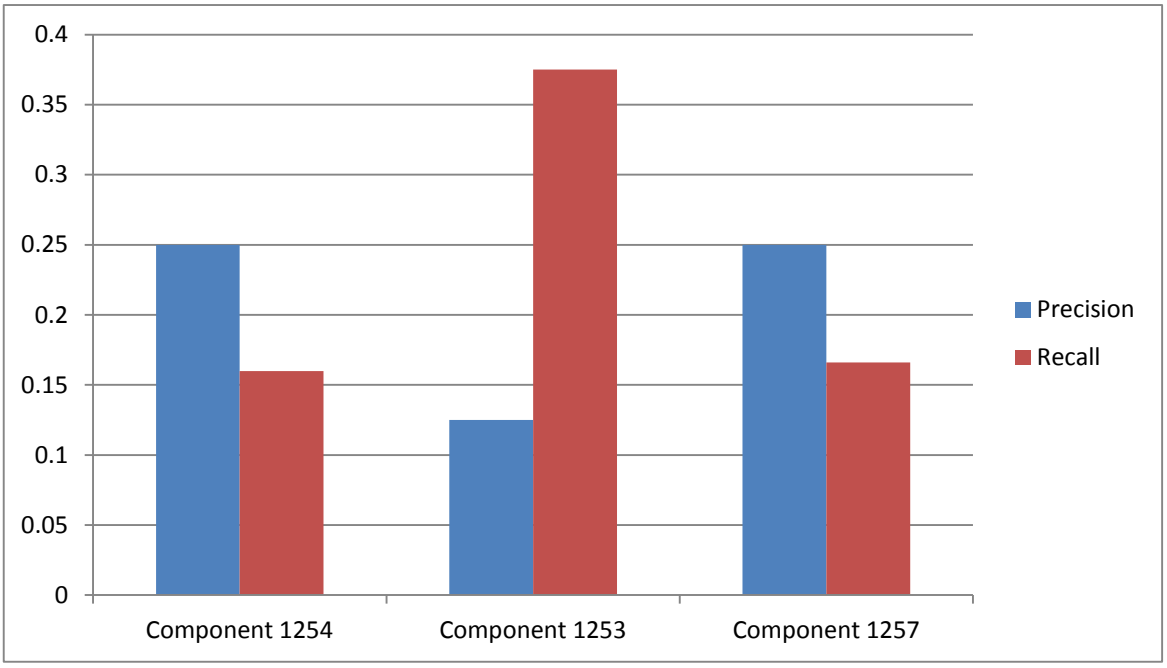

Figure 5.1: Graph shows the Precision and recall of Keyword based search

5.4.2 Result obtained by Precision and Recall ratio of Operational Semantic Based Retrieval.

The following result is obtained by the Operational Semantic Based Retrieval on the basis of precision and recall as shown in below table 5.13.

Table 5.13: Result obtained for Operational Semantic retrieval

Comp ID / Comp Criteria	**1254**	**1253**	**1257**
Precision	1	1	1
Recall	0.5	1	0.5

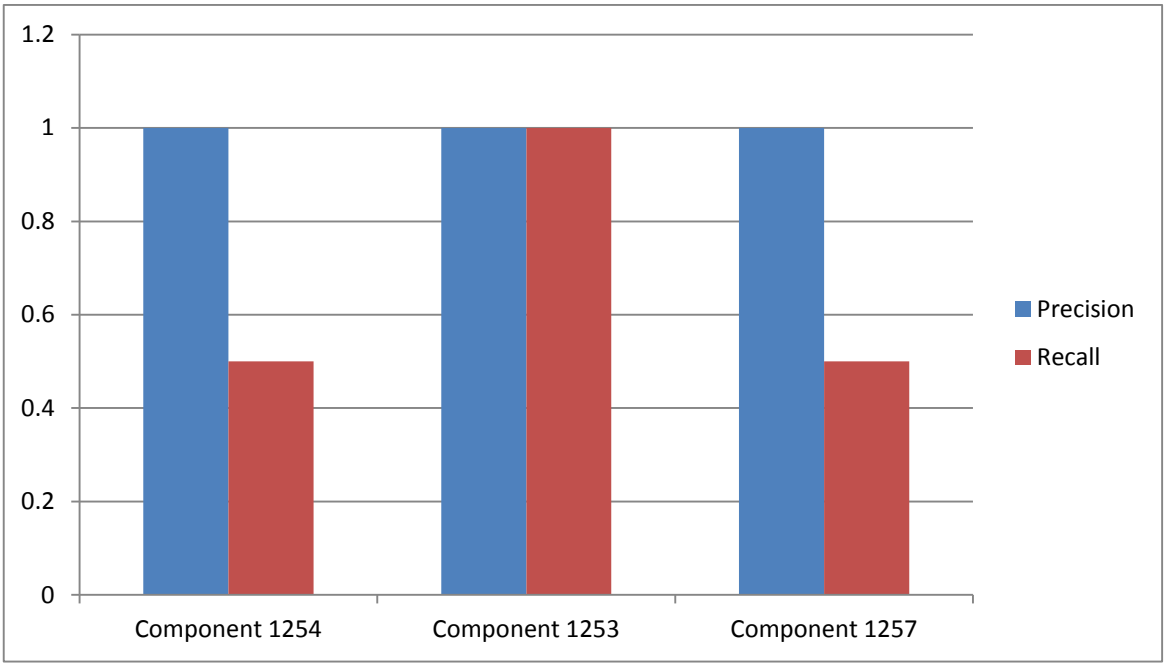

Figure 5.2: Graph shows the Precision and Recall of Operational Semantic Based Retrieval

5.4.3 Comparison between Keyword based retrieval and operational semantic based retrieval on the basis of precision and recall.

The comparison between the two retrieval techniques i.e. Keyword based retrieval and Operational semantic based retrieval show that the values obtained for Precision and Recall of the component.

For instance if we compare the values obtained for the components shown in below table 5.11, we find that the average precision of the Operational semantic based search i.e. 1. (known as perfect precision) is more than that of the Keyword based search i.e.0.208. Similarly, on comparing the values of recall we find that the value of Operational semantic based search i.e. 0.66 is more than the recall value of keyword based search i.e. 0.233 so it can be concluded that the components obtained from the operational semantic based search are more close to the user requirements than the Keyword based search.

The Comparison between both the techniques are shown in following table 5.14

Table 5.14: Result comparison on the basis of Precision and Recall

Retrieval Techniques / **Component ID**	**Keyword Based Retrieval**		**Operational Semantic Retrieval**	
			Precision	**Recall**
	Precision	**Recall**		
1254	0.25	0.16	1	0.5
1253	0.125	0.375	1	1
1257	0.25	0.166	1	0.5
Mean	**0.208**	**0.233**	**1**	**0.66**

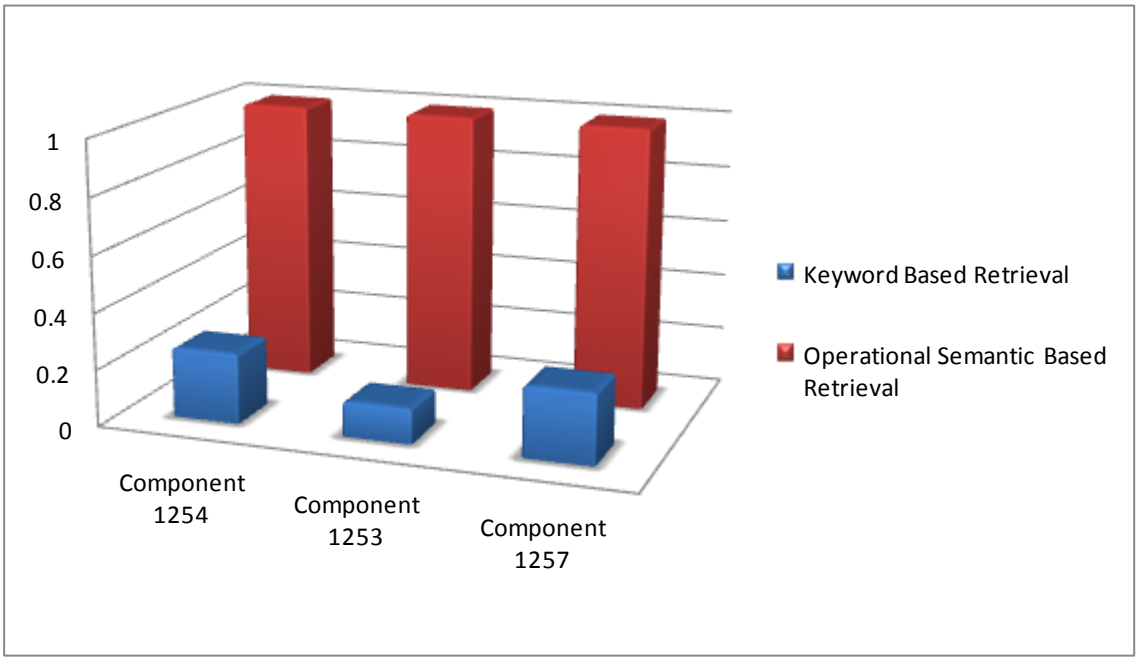

Figure 5.3: Comparison on the basis of precision between the techniques

Figure 5.3 gives the graphical representation of the above table. From Table 5.14, a graph is plotted which gives the comparison of two techniques on the basis of precision. From the graph it can be inferred that the operational semantic based retrieval has a higher precision ratio (i.e. Perfect precision) as compared to the keyword based retrieval.

Figure 5.4 gives the graphical representation of the above table. From Table 5.14, a graph is plotted which gives the comparison of two techniques on the basis of recall. From the graph it can be inferred that the operational semantic based retrieval has a higher recall ratio as compared to the keyword based retrieval.

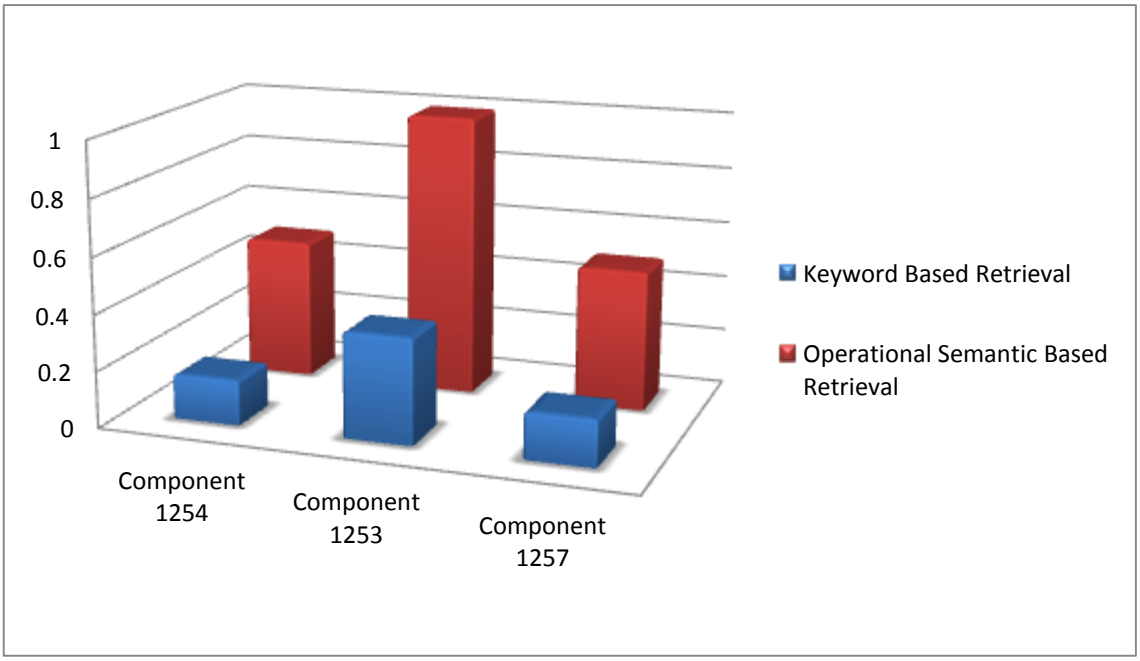

Figure 5.4: Comparison on the basis of recall between the techniques

Hence, it can be concluded that operational semantic based search is much better than keyword based search because operational semantic based search shows the best results for the exact retrieval and the precise search on the basis of precision and recall and it also saves the time and reduces effort for searching the exact match according to the need of the user requirements.

Hence, it can be concluded that the operational semantic based search is the best among all the search criteria set up in the search engine.

Results of these two techniques are more clearly shown in the following screen shots.

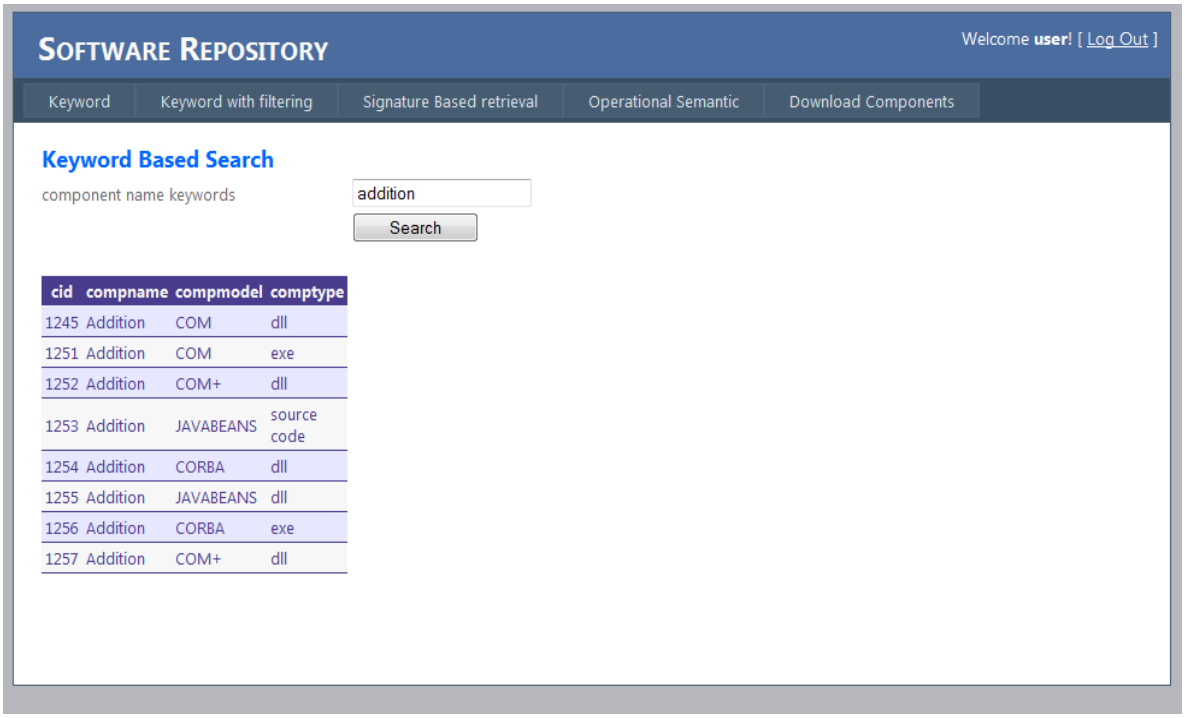

Figure 5.5: Result of keyword based search

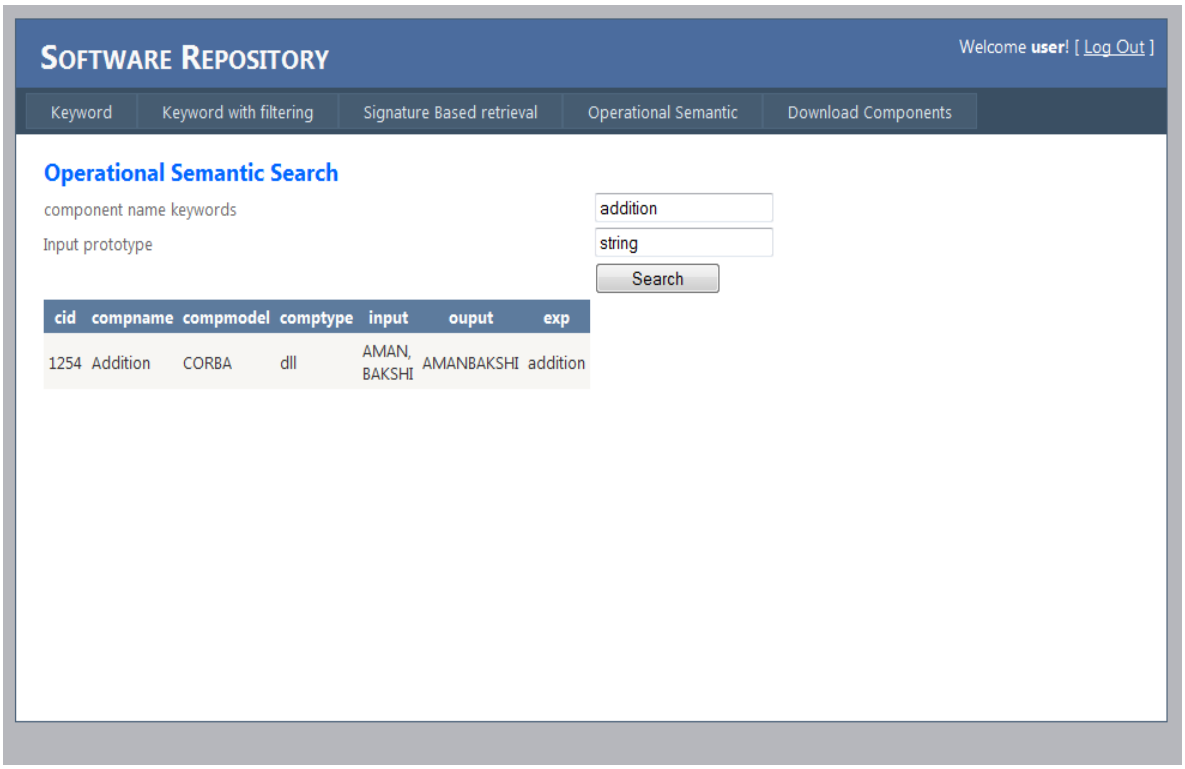

Figure 5.6: Result of operational semantic based search

Chapter 6

Conclusion and Future Scope

This chapter presents the conclusion of the work done and the future scope of the area specified.

6.1 Conclusion

This thesis presented a study on the ways to make the reusable components retrieved with more ease to the users by proposing and implementing an interface to do so. The system implemented also stores effectively the various reusable components in manageable and understandable categories from where the user can extract the desired component effectively and efficiently.

After implementation and testing of the system the conclusion come to light is that:

I. Out of all the search criteria the operational semantic based search is better than the Keyword based search because the retrieved assets are more precise and the search results are more exact.
II. Operational semantic based retrieval saves the time and reduces effort for searching the exact match according to the need of the user requirements.
III. Any user can easily download the relevant retrieved component according to his/her needs.

Thus, the system has been successfully implemented and tested conforming to the objectives set in the problem statement.

6.2 Future Scope

The Effective Retrieval can be enhanced as:

I. The repository can be extended further by adding more components in the categories present and also by adding new and relevant components in the software repository.

II. The operational semantic based search and the retrieval technique can be further converted into the semantic and syntactic based search for making the search and retrieval more efficient than the operational semantic retrieval.

III. More work is needed to improve the similarity matching algorithm to further increase the precision and recall ratio.

References

[1] W.S.Sarma and V.Rao, “A Rough–Fuzzy Approach for Retrieval off Candidate Components for Software Re- use,” Pattern Recognition Letters, Vol. 24, No. 6, 2003, pp. 875-886.

[2] P.A.González-Calero, “Applying Knowledge Modeling and Case-Based Reasoning to Software Reuse,” IEEE Proceedings – Software, Vol. 147, No. 5, October 2000, pp. 169-177.

[3] D.Lucrédio, et al., “Component Retrieval Using Metric Indexing,” IEEE International Conference on Information Reuse and Integration, Las Vegas, 8-10 November 2004, pp. 79-84.

[4] H.Mili/A.Mili, "Reuse Based Software Engineering: Techniques, Organization and Measurement", wiley-Interscience Publication, 2000, pp-9.

[5] Vanilson and Daniel, “Specification, Design and Implementation of a Reuse Repository”, In 31st annual International Computer software and Applications Conferences, IEEE, 2007.

[6] Scott Henninger, “Supporting the Construction and Evolution of Component Repositories”, In Proceedings of the 18th International Conference on Software Engineering, 1996, IEEE, pp-280-286.

[7] Jasmine Kalathipparambil and vasantha, “A Mixed Method Approach for Efficient component retrieval from a component repository”, Journal of Software Engineering and Applications, 2011.

[8] Navjoti Pandhi, “reusing components using genetic components”, ME thesis, May 2004.

[9] A.Mili, R.Mili and R.T.mittermeir, “A Survey of Software Reuse Libraries”, Annals of Software Engineering 5, 1998, pp-349-414.

[10] Pfister, Joachim and Hans-Dieter Zimmermann, “Towards the Introduction of an Institutional Repository: Basic Principles and Concepts”, 2008, pp- 230.

[11] John Grundy, “Storage and Retrieval of Software Components using Aspects”, IEEE, 2001.

[12] Hanen Ben Khalifa, Oualid Khayati, Henda Hajjami Ben Ghezala, “A Behavioral and Structural Components Retrieval Technique for Software Reuse”, IEEE Computer Society, 2008, pp-134.

[13] Yumen Ye, “An Active and Adaptive Reuse Repository System”, 34th Hawaii International Conference on System Sciences, IEEE, 2001.

[14] John Grundy, “Storage and Retrieval of Software Components using Aspects”, IEEE, 2001.

[15] Jihyun Lee, Jinsam Kim and Gyu-Sang Shin, “Facilitating Reuse of Software Components using Repository Technology”, 10th Asia-Pacific Software Engineering Conference, IEEE, 2003.

[16] Ronaldo C.Veras and Silvi.R.L.meira, “Comparative Study of Clustering Techniques for the Organisation of Software Repositories”, IEEE, 2007.

[17] Hanen Ben Khalifa, Oualid Khayati, Henda Hajjami Ben Ghezala, “A Behavioral and Structural Components Retrieval Technique for Software Reuse”, IEEE Computer Society, 2008, pp-134.

[18] Oualid Khayati, Jean-Pierre Giraudin, “Components Retrieval Systems”, 2001, pp-56.

[19] M.R. Girardi and B.Ibrahim, “Automatic Indexing of Software Artifacts”, 1994.

[20] Jingbo Zhu, Tianshun Yao: “A Knowledge-based Approach to Text Classification”, IEEE, 2001.

[21] Divya Pandove, “Designing an interface for effective retrieval of components”, ME thesis, June 2010.

[22] Panos Constantopoulus and Martin Dorr, “Component Classification in the Software Informational Base”, Prentice Hall, 1995, pp-177-200.

[23] Daniel Lucredio, “A survey on Software Components Search and Retrieval”, 30th EUROMICRO conference, IEEE, 2004.

[24] Y.S.Maarek, D.M.Berry and G.E. Kaiser, “An Information Retrieval approach for automatically constructing software libraries”, IEEE transactions on software engineering, 1991.

[25] S. Henninger, "An evolutionary approach to constructing effective software reuse repositories," ACM Transactions on Software Engineering and Methodology, vol.6, 1997.

[26] R.Prieto-Dı´Az, and P.Freeman, “Classifying Software for Reusability”, IEEE Software 4, 1987, pp- 6–16.

[27] Wang Haitao and Chen Xing, “Study on a Component Library Model Based on the Four-Layer Architecture”, International conference on intelligent networks and intelligent systems, IEEE, 2011.

[28] Zhu Jingbo, Yao Tianshun, “A Knowledge-Based Approach To Text Classification”, IEEE, 2001.

[29] Chao-Tsun Chang, William C. Chu, Chung-Shyan Liu, Hongji Yang, “ A formal approach to software components classification and retrieval,” 264-269 Electronic Edition, IEEE Computer Society DL, Colin J. Hardy, Hlen M. Edwards, J. Barrie Thompson, 1997.

[30] Lei Zhang, LichaonChen, Lihu Pan, Yingjun Zhang, “A Novel Approach of Component Retrieval in Large-Scale Component Repositories”, IEEE, 2012.

[31] Cem Kaner, “Quality Assurance in software testing”, Orlando, November 2006.

[32] Jiang Guo and Luqi, “A Survey of Software Reuse Repositories”, IEEE, 1999.

[33] GE Junwei, TAO Cong, FANG Yiqiu, “Architecture for Component Library Retrieval on the Cloud”, IEEE, 2011, pp 536-539.

[34] P. Chen, R. Hennicker, and M. Jarke," On the retrieval of reusable software components", August 1997.

[35] Pietro Abate, Roberto Di Cosmo, “Learning from the future of Component Repositories”, ACM Transactions on Software Engineering, 2012, pp-51.

List of Publications

[1] Amandeep Bakshi, Seema Bawa, "A Survey for Effective Search and Retrieval of Components from Software Repositories", International Journal of Engineering Research & Technology, Vol. 2, Issue 4, April 2013, ISSN: 2278-0181, pp-1935-1939.

[2] Amandeep Bakshi, Seema Bawa, "Development of a Software Repository for the Precise Search and Exact Retrieval of the Components", communicated to International Journal of Advanced Research in Computer Science and Software Engineering, July 2013.