DESIGNING AN INTERFACE FOR EFFECTIVE RETRIEVAL OF COMPONENTS

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

Master of Engineering

in

Software Engineering

By:

Divya Pandove

(800831028)

Under the supervision of:

Mrs. Shivani Goel

Assistant Professor

Computer Science and Engineering Department

Thapar University, Patiala.

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2010

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Designing an Interface for Effective Retrieval of Components"**, in partial fulfillment of the requirements 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 **Mrs. Shivani Goel**, Asst. Professor, Computer Science and Engineering Department, Thapar University, Patiala and refers other researcher's works which are duly listed in the reference section.

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

Divya Pandove

(Divya Pandove)

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

Shivani Goel 30/6/10

(Mrs. Shivani Goel)
Computer Science and Engineering Department,
Thapar University,
Patiala.

Countersigned by

30/06/10

(RAJESH BHATIA)
Head,
Computer Science and Engineering Department,
Thapar University,
Patiala.

R.K.Sharma 1/7/10

(R.K.SHARMA)
Dean (Academic Affairs)
Thapar University,
Patiala.

Acknowledgement

First of all, I want to thank Almighty, who has always guided me to work on the right path of life. My greatest thanks to my parents who bestowed the ability and strength needed to complete this work on me. I am deeply indebted to my friends Amandeep , Rupinder , Gagandeep and Jarrar for their ever encouraging moral support and help, which enabled me to pursue my studies.

This work would not have been possible without the encouragement and able guidance of my supervisor, **Mrs. Shivani Goel,** Asst. Professor, Computer Science and Engineering Department, Thapar University, Patiala. Her enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions founded in this thesis are a result of our various stimulating discussion sessions. Her feedback and editorial comments were also valueable for writing the thesis.

Also, I shall be failing in my duties if I do not thank **Dr. Rajesh Bhatia**, Head, Computer Science and Engineering Department, Thapar University, Patiala, as he has been a constant source of inspiration for me throughout this work. He is the person who introduced me to this area of work and inculcated the interest in me to carry forward my thesis research in it.

I would also like to express my sincere gratitude towards the entire faculty and staff of the Computer Science & Engineering Department, for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Divya Pandove

Divya Pandove
(800831028)

Abstract

Every software component is prepared for some specific purpose but there may be some specific features in it, which can be reused for some other purposes as well. The problem that is generally encountered while doing so, is the efficient search and retrieval of these components as they are not properly represented. So the basic task of the thesis 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 interface which provides for this and coming up with the best solution for reorganization of the components in the repository so that, there is maximum recall, coverage ratio and precision 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.

In order to introduce black box and white box reuse, the components have been divided into seven basic reuse categories and a provision for adding new categories has also been provided to the Administrator. The user can perform search, either based on these categories and sub categories or can do it on the basis of typing the keywords and signatures of the desired components or can select the component from the entire list of the displayed components.

The basic comparison between the three techniques on the basis of precision, recall and coverage ratio gives the better way to do so among all three of them.

Thus, the thesis deals with evaluating the best technique for retrieval of the software components for the purpose of reuse.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Software Reuse

Software Reuse is the process of implementing or updating software systems using existing software assets [1]. Although it is believed that "Software Asset" is simply another term for source code, this is not the case. Software assets, or components, include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites [2]. Anything that is produced from a software development effort can potentially be reused.

1.2 What is a Reuse Library?

A reuse library is simply a repository that stores reusable components and must have the characterisation of the assets that are included within. In order to make effective use of a reuse library 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 [3].

1.3 What are Reusable Assets?

The most common form of reusable artifact is the source code in some programming language, but it is not the only one [4]. There are some other commonly used reusable assets, which are:

- **Executable Code**: The essence of a piece of executable code is function that it computes; executable code is typically represented in machine readable form and is indexed by means of its functional properties.

- **Source Code**: Much of what is true is true for source code, to the extent that source code embodies a function. But to the extent that it also embodies structural information, source code can be viewed as problem solving knowledge. Source code

is of course, represented by programming languages and can be indexed by means of its structural as well as functional properties.

- **Requirement Specifications**: Whereas code assets are executable, requirement specifications are not; rather, they are the products of eliciting user requirements and recording them in some notion. Specifications can be represented in natural language, in formal notation (logic, axiomatic systems, formal languages), or in a mixture thereof. Specifications are indexed by means of the functional properties they capture, and may be reused to build either compound specifications or (after modifications) variations on the original product.

- **Design**: Designs are generic representations of design decisions; their essence is the design/problem – solving knowledge that they capture. In contrast to code assets, design is not executable; in contrast to specification assets, they capture structural information rather than functional information. They are represented by patterns that can be instantiated in different ways to produce concrete designs. Unlike functions or modules, designs cannot be indexed by their functional properties; rather they can be indexed by features of the family of problems they solve.

- **Test Data**: Imagine that we have developed software and we have integration – tested this system using some test data. We may want to reuse the same data to test the system subsequently, following a maintenance operation; also we may want to reuse the test data to test a similar product, which has similar set of inputs but different output conditions (for example, that the same test data can be used to test a procedure that sorts arrays in increasing order, and a procedure that sorts arrays in decreasing order). For this reason, test data are a perfectly legitimate reusable asset. Representation of these data is straightforward, and that data can be indexed by description of the input domain of the software system, or possibly some general indication of the function of the system.

- **Documentation**: Natural-language documentation that accompanies a reusable asset can conceivably be considered as a reusable asset itself; in addition, many reusable

assets (specifications, design) can be represented as natural-language documentation. Documentation is most typically represented in natural language and can be indexed via the asset that it documents; other indexing mechanisms for documentation include information retrieval techniques and tools, as well as hypertext tools.

- **Architecture**: Software architecture defines the structure of a software system as the aggregate of a set of components that exchange data. The constructs by which building blocks are usually combined in architecture have higher levels of abstraction than do programming language constructs, and are of different nature; they prescribe information flow, control flow or communication protocols between components. Architectures are represented by means of specialized notations, and are indexed by means of their architectural features.

1.4 Introduction to Component Storage and Retrieval

The reuse of software assets is increasing rapidly with the expansion of software industry. A reusable software component is self contained, clearly identifiable piece that describes and / or performs a specific function, have clear interfaces, appropriate documentation and a defined reuse status [5].

The most important requirement that rises with the existence of these assets is the presence of a storage structure or a repository that would contain these assets and also provide for an easy search and retrieval of the same. The repository has to be evolutionarily constructed and maintained in order to maximize the use of reusable assets. 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 user's 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.

Each and every aspect starting from storage of assets to their retrieval are faced with some major challenges and prospects that need to be addressed in order to make a efficient storage and retrieval system, which will maximize the use of reusable components, hence, reducing the effort, time and other resources that go into building them again and again.

1.5 Assessment Criteria for Storage and Retrieval Methods

The basic criteria for comparing the various storage and retrieval methods are [6]:

- **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.

- **Coverage Ratio**: The coverage ratio of a retrieval algorithm is the average number of assets that are visited over the total size of the library. This number ranges between 0 and 1. The brute force exhaustive navigation produces coverage ratio of 1, but more elaborate algorithms may take advantage of the storage structure to exclude portions of the library from consideration without affecting recall. We may use the five point scale instead of the numeric scale here as well.

  The aim of any search program is to maximize precision, recall and coverage ratio, so as to provide Effective Retrieval.

1.6 Organization of Thesis

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 'Effective Retrieval'. It defines the basic features of the system along with construction of the

repository and also establishes the various methods of search and retrieval of the components. It further goes on to explain the working of the system.

The fifth chapter deals with the experimental results obtained of the system on the basis of precision, recall and coverage ratio.

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

Chapter 2

Literature Survey

2.1 Creating a Repository

Any repository, may or may not supporting reuse, will be constructed by following a common procedure for its development [7]:

- **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 emphasise 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 organisation 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 organisation. 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.

2.2 An Evolutionary Approach to Constructing Effective Software Reuse Libraries

Repositories for software reuse are faced with two interrelated problems:

- Acquiring the knowledge to initially construct the repository.
- Modifying the repository to meet the evolving and dynamic needs of software development organizations.

To deal with both these problems, an evolutionary construction of repositories must take place. An example of this type of repository is, The Code finder – PEEL repository [8].

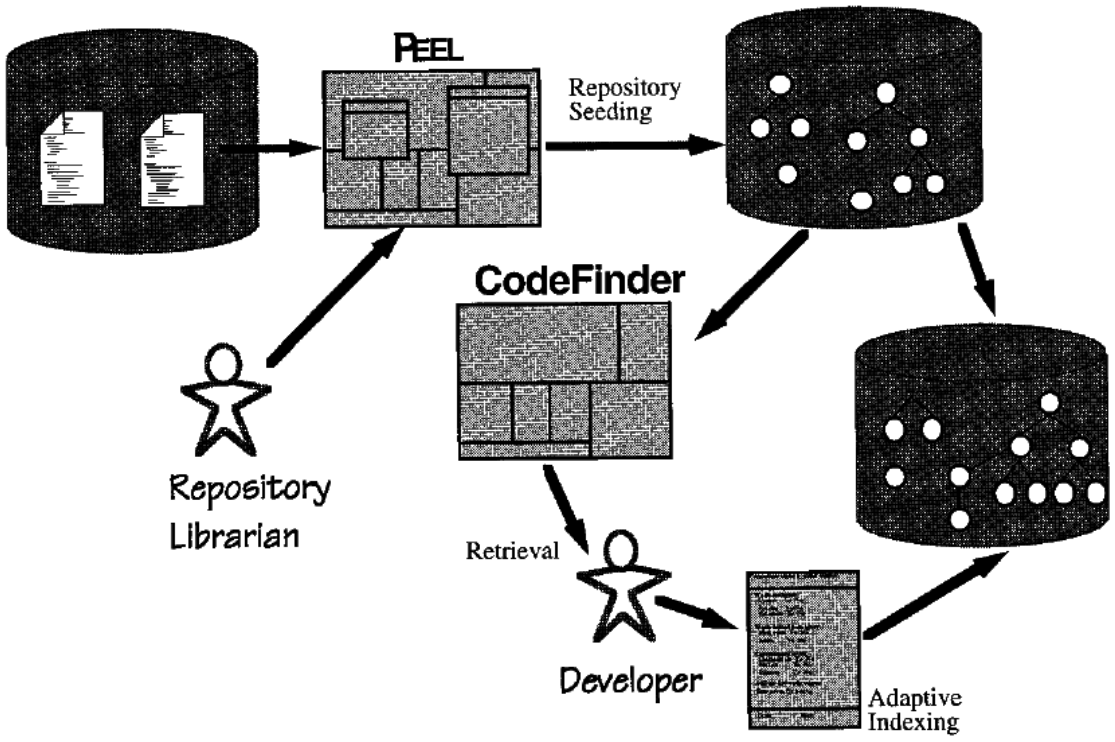

Figure 2.1: Code Finder PEEL Repository [8]

Figure 2.1, shows how repositories can be constructed with minimal up-front effort and then incrementally improved as the repository is used. In the initial stages of constructing a repository, it is important to populate the repository with components, even if the components are not explicitly designed for reuse, this is called seeding [9].

The prototype system, called PEEL (**P**arse and **E**xtract **E**macs **L**isp), extracts components from text files and indexes them through a combination of automatic extraction and interactive user support. The Code Finder system is a prototype retrieval tool that uses a combination of retrieval techniques to help users find reusable software components [10].

2.2.1 Seeding the Repository

Seeding a repository is a matter of creating component representations, by indexing them with key terms and phrases. Components can take on any size or form, depending on the needs of repository users.

2.2.2 Procedure Used by PEEL

PEEL uses a three-step procedure to process terms:

- The first step is fully automated. Terms are extracted from the name of the function and any strings or comments within or immediately preceding the definition. A common stop list is applied to the extracted terms to remove common words with little semantic meaning such as “and,” “of,” “the,” and others [11]. PEEL users can also augment this stop list or add terms to a nonsense word list through the operations “Put this term on the Stop Word list” and “Put this term on the Nonsense Word list”.
- The second step is to display the terms to the user. Users can remove the terms, truncate the list (a feature that proved to be useful for many components in which only the name was descriptive), and highlight the term in the source code window, to show how and where the term has been used.
- The third step allows users to add their own terms, which can include phrases (words separated by spaces) and any other punctuation.

2.2.3 Code Finder

Code Finder also creates hyperlinks to all functions called by a component. These functions are displayed in mouse-sensitive text in the Code Finder Interface. Displaying the immediate Sub system of a component is therefore a mouse click away.

2.2.4 Challenges Faced in Repository Construction

We are faced with a number of challenges when it comes to repository construction for reusable components, some of them are:

- 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.
- 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.
- 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.
- 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.

2.3 Information Management Process

Information is basically of two forms:

- Structured information.
- Unstructured information.

Structured information is stored in databases with predefined structure to cater to the information needs, unstructured information is present in form of bitmaps or textual data and does not confirm to any data type definition.

In order to have seamless access to information, the information needs to be digitalised, normalised and archived.

To seed the repository, the information has to be converted into assets. The sequence to achieve this transformation is [12]:

- **Assertize Information:** A lot of information is present in any organisation. This information, like any other physical asset, has the ability to provide returns for the enterprise over their long span. Some governing rules would identify as to which information will become an asset. These governing rules would be decided on the basis of:

  - Originator of the information.
  - Strategic needs of the enterprise.
  - Types of information
    - ✓ Client testimonial.
    - ✓ Customer feedback.
    - ✓ E-learning experience.
    - ✓ Proposal presentation.
    - ✓ Marketing collaterals.

- **Digitize the Asset (existing in non digital format):** The process of digitization involves scanning documents and converting them into standard file formats like pdf. These documents are amendable for creation of indexes and search.
- **Add Metadata Vocabularies:** Once the asset is digitalised, then it is marked with Meta tags describing the content. A central Meta tag managed repository replaces older versions with newer ones.

  An automatic enveloping takes place after this, which refers to process of defining structures

  for easy search and retrieval of content.

  Unstructured Information + Meta Tag = Enveloped Asset.

  Thus, by the combination of indexing tools and a metadata assigning workflow, the Information repository is created.
- **Store Information Asset**: The information asset is stored in ‘rights managed software library’. Storage of the digital asset is designed for automated backup redundancies and high availability. The metadata and content are stored in separate

repositories. The databases also have access rights based on the role of the person trying to access the assets.

- **Update Tags:** As the assets get used, they are rated according to the relevance of the asset to the present needs of the organisation and the accuracy of the metadata in describing the asset it envelops are assigned. Over time as the usage of asset and rating of its relevancy mature, the updation of metadata becomes significant.
- **Remove Assets**: With time the digital assets become obsolete. The items for removal that can be removed from the information repository could be identified by metadata search for discontinued products or business and by low rating in the user feedback for asset effectiveness.

By this process of continuous addition and updation, the value of an information asset keeps improving and is made easy to access and use.

Figure 2.2, shows the Information Management Process. The digitalised assets are identified and tagged, then Meta data is added to it and then they are stored in the Information Management System.

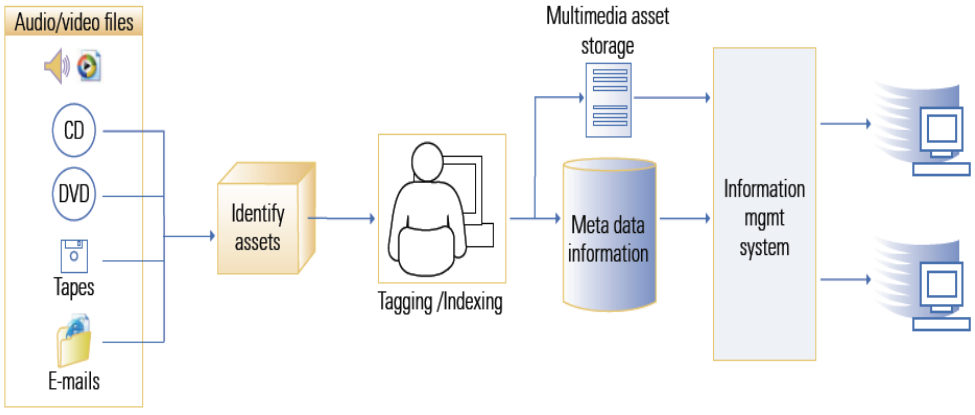

Figure 2.2: Information Management Process [12]

2.3.1 Challenges Faced in Information Management

The major issues that come in picture while managing the information are:

- Low reuse of Information – The resources spent on converting the information into assets may be much more than the profits gained by reuse of these assets.
- Inadequate Information security – A lot of information exists as a result, it may cause inability to define as to who is authorised to access what information.
- Lack of version control – The cost increases with the introduction of version metadata maintenance but it not implemented it would result in inconsistent and outdated assets.
- Lack of data protection – When information becomes an asset, in that case it also needs to be protected against loss and corruption as a result extra resources need to spent on constructing a backup.
- Redundancy – The information comes from various sources as a result there is a chance that it can become redundant.
- Inefficient search and retrieval of information.

2.4 Problem Faced in Searching Reuse Library

When faced to a problem, the re-user understands it in his own manner. In order to find a component that suits to this problem, he formulates a query, which may be as simple as a keyword set, or as complex as specifications in a formal language [13].

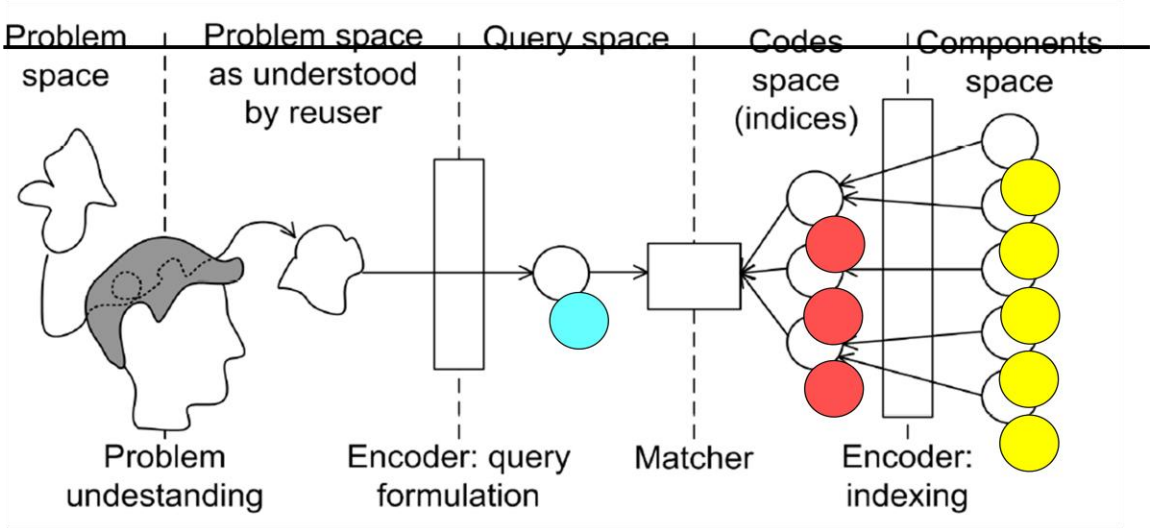

Figure 2.3: Component search problem [13]

Figure 2.3, shows that the above process results in information loss, since the reuser is not always capable of correctly understanding the problem, or to adequately formulate the query.

To allow the components to be automatically searched, their information must be coded, manually or automatically, in a process known as indexing, or classification. This process of producing indexes also results in information loss. Search itself consists in matching the formulated query with the indexes.

Most research in component search aims at reducing the information loss:

- In the problem understanding.
- In the query formulation.
- In the indexing process.

In case of exact match, the retrieved component becomes a candidate of the black box reuse and in case of approximate retrieval, the component has to be altered and is a candidate of white box reuse.

The most important issue is that of the indexing or the characterization of the assets.

2.4.1 Challenges Faced While Searching a Component

There are different levels of knowledge that the re-user may have in relation to a components repository. Some elements are well-known, others are vaguely known, some are believed to exist and others are totally unknown [13]. When faced to a problem, the re-user usually searches for elements that are well-known and vaguely known. Other elements are ignored, and the reuse opportunity is lost.

2.5 Component Indexing Process

The structure of a repository is key to obtaining good retrieval results. No matter how "intelligent" the matching algorithm, if components are indexed or otherwise structured poorly it will be difficult to achieve good retrieval performance [8].
Some basic indexing techniques are:

- Enumerated classification.
- Faceted classification.
- Free text indexing.

2.5.1 Enumerated Classification

It is a well-known retrieval method used by the Dewey Decimal system and ACM's Computing Reviews Classification System. In this method, information is placed in categories that are usually structured in a hierarchy of sub categories; the appeal of the classification scheme is the ability to iteratively divide an information space into smaller pieces that reduces the amount of information that needs to be perused [8].

2.5.1.2 Challenges Faced in Enumerated Classification

- It requires users to understand the structure and contents of repository to effectively retrieve the information.

- There is inherent inflexibility present in this scheme.
- Once the hierarchy is in place, it gives only one view of the repository. Changes to that view may reverberate throughout the taxonomy, resulting in extensive redesign of class structures.
- There is trouble in distinguishing between class labels such as Proceedings and Inproceedings in a database of computer science literature references [14].

2.5.2 Faceted Classification

Faceted classification avoids the enumeration of component definitions in a hierarchy by defining attribute classes that can be instantiated with different terms [15]. Terms are grouped into a fixed number of mutually exclusive facets. 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. Faceted techniques use a fixed number of facets (attributes) per domain. Facets are more flexible than enumerated schemes because individual Facets can be redesigned without impact on other facets. Facets make it easy to synthesize and combine terms to represent components.

2.5.2.1 Challenges Faced in Faceted Classification

- It becomes hard for users to find the right combination of terms that accurately describe the information need, especially in large or complex information spaces [16].
- The method also requires that users know how the library and terms are structured and have an understanding of the significance of each facet and the terms that are used in the facet [17].
- Field use of faceted retrieval systems has shown the need for training of people to use facets effectively, and even more extensive training is necessary for designing faceted information domains [18].

2.5.3 Free Text Indexing

Free-text indexing (automatic indexing) methods use the text from a document for indexing. Document text is applied to a "stop list" to remove frequently occurring words such as "and" and "the." The remaining text is used as an index to the document. Users specify a query using keywords that are applied to the indices to find matching documents. No classification effort is required, although human indexers are sometimes used to augment automatically extracted index terms. Matching criteria can range from Boolean match to more sophisticated methods, such as the vector model, that use statistical measures to rank retrieved information [19]. Free-text methods are simple to build and retrieve from, but rely on regularities in linguistic texts that need large bodies of text to become statistically accurate. The non linguistic nature of source code and the fact that clear and accurate documentation is not necessary for working code make these methods less attractive for software component repositories than for text documents. Free-text methods are most applicable to domains with extensive documentation [20].

2.5.3.1 Challenges Faced in Free Text Indexing

- It would be inaccurate to characterize most source code as being documented adequately for these methods.
- Retrieval effectiveness of free-text methods has been questioned within text-intensive domains [21].
- The vocabulary becomes limited, freezing chances of innovation by the user.

2.6 Component Retrieval Process

In the retrieval process, the Information retrieval system extracts the documents or more generally, the pieces of information which will presumably answer the information need formulated by the user. This retrieval process is usually separated into a preliminary step of indexing the documents, followed by an operational step of retrieval. The retrieval process can be iterative, if the user provides some feedback on the retrieval results.

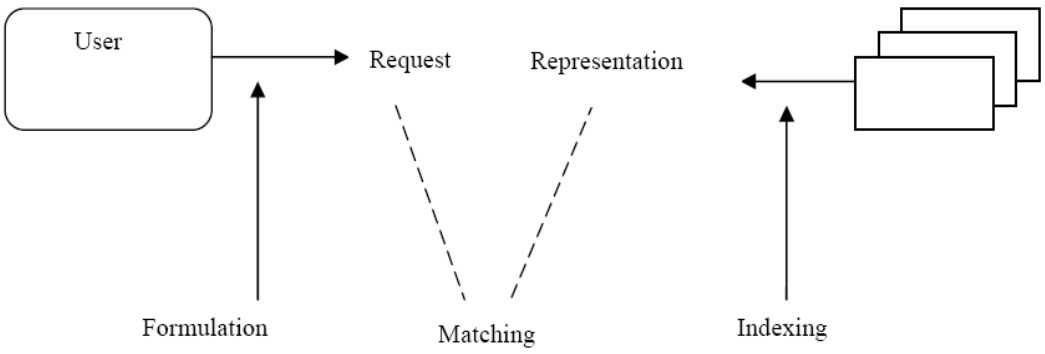

Figure 2.4: A General View of Retrieval Process [22]

Figure 2.4, shows the steps involved in the retrieval process, Information Retrieval System has three basic issues, these are:

- The formulation of the query.
- The indexing process.
- The matching function.

2.6.1 Formulation of Query

The problem of the formulation of a query is closely related to the issue of information need.

Information need is distinguished into two types:-

- The extensional information need.
- The intensional information need.

An extensional information need is a clear and specific information need.

An intensional information need is much less clear and specific.

2.6.2 Indexing Process

Information must first be stored in some way in the system. In general, it is unlikely that the full text of each document is stored in the system, especially if a large document

collection is concerned. Therefore, each document is assigned a descriptor, which is a representation of its informative content within the system. The process by means of which such a representation is generated is called indexing. Indexing can be done manually or automatically.

2.7 Matching and Relevance

Once the user has formulated his information need into a request, this query is compared with the document representations. This comparison is called matching in Information Retrieval.

The distinction is made between exact matches and partial matches. In case of an exact match, the query representation and the document descriptor must be equal in order to retrieve the document. In case of a partial match, the document is retrieved when the representation of the document is similar enough to the representation of a query. When a document is ‘similar enough’ to a query, it is ‘relevant’ with respect to that query. Relevance is a very important notion in Information Retrieval, since the main purpose of Information Retrieval is to retrieve as many relevant documents as possible in response to a user request.

2.8 Challenges Faced in Retrieval Process

The most prominent challenge faced in the retrieval process is the correct formulation of the query according to the indexing technique used to represent the assets. As only an appropriate query will result into a correct match and the relevance of the asset fetched would also be high.

2.9 Main Approaches for Software Classification and Retrieval

Many software classification and retrieval techniques are available. Some of them are discussed as follows:

2.9.1 Formal Specifications

Using formal specifications to represent software components facilitates the determination of reusability because they more precisely characterize the functionality of the software, and the well-defined syntax makes processing amenable to automation.

Formal specifications use mathematical notation to describe in a precise way the properties, which an information system must have, without unduly constraining the way in which these properties are achieved.

They describe what the system must do without saying how it is to be done. This abstraction makes formal specifications useful in the process of developing a computer system, because they allow questions about what the system does to be answered confidently, without the need to disentangle the information from a mass of detailed program code, or to speculate about the meaning phrase in an imprecisely worded prose description [23, 24].

In this approach, queries are formal requirement specifications (e.g., the specification of a signature) and the system retrieves relevant software from a library of formally specified components by invoking a theorem prover to determine if component specifications satisfy the requirements.

2.9.2 Challenges Faced in Implementing Formal Specifications

- The biggest challenge in this approach is to understand and implement mathematical functions and concepts.
- An appropriate notion has to be chosen from the wide variety of formal specification languages available.
- The cost of development shall increase many folds.

2.10 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 [25].

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. The principal process for the Knowledge –based text classification is illustrated in Figure 2.5:

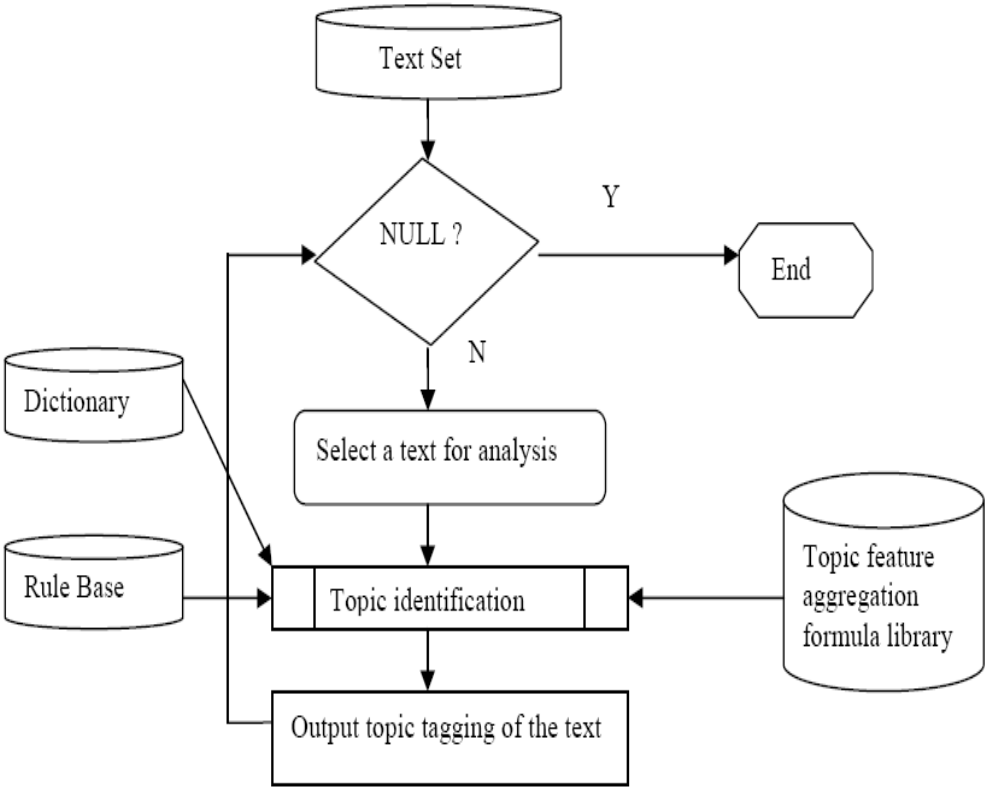

Figure 2.5: Knowledge Based Text Classification [25]

2.10.1 Challenges Faced in implementing Knowledge Based Approach

The major challenge in this system is that, it usually requires enormous human resources as Knowledge bases are created for each application domain and are usually populated manually.

2.11 Hypertext

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. Figure 2.6, shows the structure of hypertext technique.

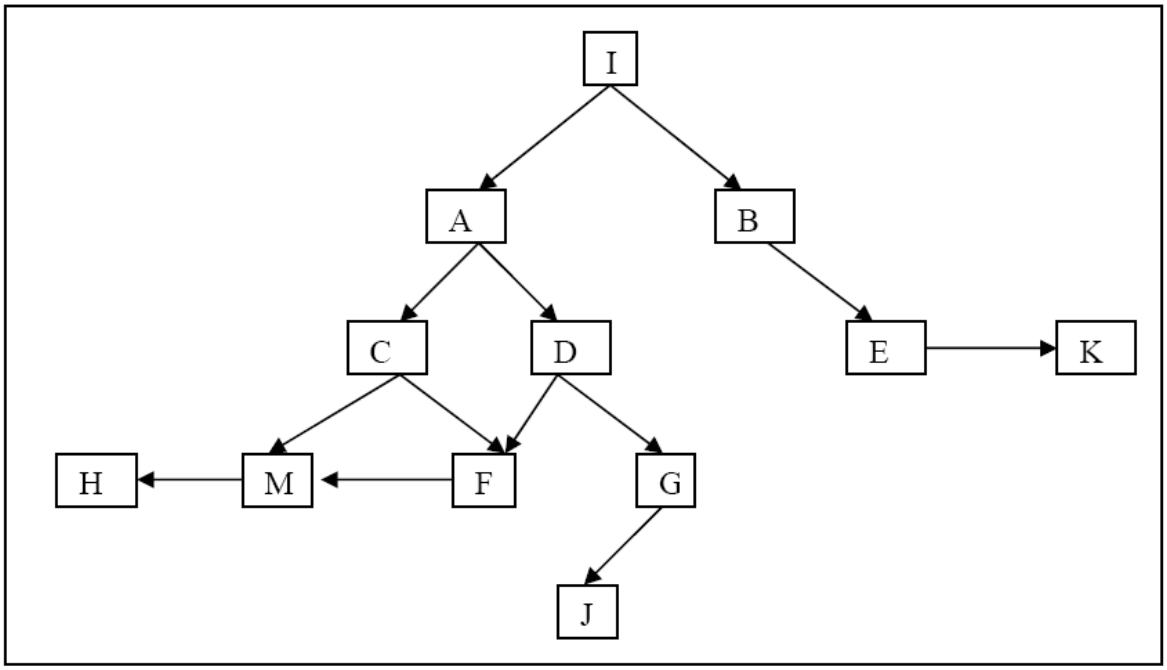

Figure 2.6: Hypertext structure [26]

2.11.1 Challenges Faced in Implementing Hypertext Approach

- The relationships between various nodes should be predefined.
- In case of large number of hyperlinks referring to same asset, the links get cluttered.

2.12 Browsing

Browsing is an example of a retrieval technique 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 [27].

2.12.1 Challenges Faced In Implementing Browsing Approach

- The users should have the ability to position themselves in an area of interest of the database.
- The user should have the ability to be able to recognize appropriate directions in which to further the search.
- The user should have the ability to easily move quickly and efficiently through the database.

2.13 Behavioural Based Retrieval

The behavioural 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 behaviour. An ordering on behaviours is then used to classify components and to search through the library of components. The programs used to produce the components behaviour 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 behaviour. The components that respond to the searched behaviour will be selected and presented to the user[28, 29].

2.13.1 Challenges Faced in Implementing Behavioural Based Approach

- The user should have a thorough knowledge of the language in which query has to be written.
- A lot of time and resources need to be spent on the study of behaviour and functions of each component.

2.14 Automatic Indexing

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 [30].

2.14.1 Challenges Faced in Automatic Indexing Approach

The biggest challenge in the automatic indexing is of the limited vocabulary of the index and the user has to search using only that vocabulary.

This chapter summarizes various approaches and challenges in setting up a repository; the next chapter introduces the problem taken up in the thesis.

Chapter 3

Problem Statement

Every software component is prepared for some specific purpose, but there may be some specific features in it, which can be reused for some other purpose as well. The problem that is generally encountered while doing so is the efficient search and retrieval of these components as they are not properly represented. So, the 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 interface providing for the same and to come up with the best solution for organization of the components in the repository so that, there is maximum recall, coverage ratio and precision 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 softwares 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 either based on certain fixed categories and sub categories or can do it on the basis of typing the keywords and signatures of the desired components or can select the component from the entire list of the displayed components.

The basic comparison between the three techniques on the basis of precision, recall and coverage ratio gives the better way to do so among all three of them.

Thus, the thesis deals with evaluating the best technique for retrieval of the software components for the purpose of reuse.

Based on the above problem statement we have set the following Objectives:-

- Identifying the basic search and retrieval problems faced by the users.
- Development of a repository using the evolutionary approach.
- Indexing of the components of the repository.
- Construction of an interface that facilitates easy search and retrieval of these components.
- Implementing various retrieval techniques.
- Comparing these techniques on the basis of some fixed criteria to find out the one best suited for efficient retrieval.

Chapter 4

Design and Implementation of Effective Retrieval System

The newly developed system is named as Effective Retrieval. It is a web application developed in ASP.Net with SQL server at the backend. This chapter covers the features, architecture, design and implementation of the Effective Retrieval system. The evolution of the system deals with construction of a reuse repository and systematic storage of components. The system further facilitates the efficient and appropriate retrieval of these components. The last part of this chapter contains the working of the system along with its screen shots.

4.1 Features of the Effective Retrieval System

The main features of the proposed system are:

- It contains a dedicated repository for storage of components that can be reused.
- Each component has a separate surrogate which represents it in the repository.
- It contains an interface, which provides with the various categories and subcategories of reusable components for the user to confine his search.
- The interface also provides with the keyword based search, in which the user is free to search on his own will based on the type of surrogate he prefers to search from.
- The system implements the use of a dynamic surrogate for each type of search and retrieval mechanism, which the surrogates of the components keep on changing according to the choice of the user.
- It also implements version control, wherein all the versions of a component are saved and displayed separately.
- A separate administrative unit is also present; in order to manage various categories, subcategories and user added criteria.

4.2 Construction of Repository

This section introduces to the black and white box reuse of components then it goes on to explain the representation of the assets in the repository of "The Effective Retrieval System". Further the steps followed to construct the repository are enlisted.

4.2.1 Black box and White Box Reuse of Components

The basic unit of the repository is the component. Now, the components in terms of reuse can be defined in two ways:

- Black Box Reuse Components: These components can be realized into the target system without any modifications. Though these components are of greater use to the user but they are difficult to realize. The COTS (components of the shelf) is an example of such kind of reuse. So, the repository must contain some components like COTS so as to increase the usability of the system. These are in executable form *i.e.* no source code is available.
- White Box Reuse of Components: These components need to be modified before integrating them into the target system. In order to do so, one requires a great understanding of the inner working of the component. This type of reuse may bring smaller benefits but are easy to realize. The repository must also contain the assets which are easy to modify and reuse *i.e.* the source code of the components is also stored.

4.2.2 Representation of the Components

The components in the repository are represented by using a surrogate. A surrogate may be defined as the metadata which describes the component almost fully. The surrogate in the Effective Retrieval system contains the following fields:

Title: Here the main heading of the component in entered.

Description: The component is textually defined here. A detailed summary of the component is expected to be entered in this field.

Features: The main features of the component are listed here. This column represents the functional aspect of the component.

Version: This column maintains and controls the various versions of the same component.

Signature: If the present component has a particular signature then it is entered in this column. This signature can later on be used to perform a constrained search based on this signature only.

Category: A dropped down menu containing the basic categories is provided. If the component being added belongs to any one of the present categories, then the category is added and if it belongs to a new category, then that category needs to be specified. This category can later be made of the basic category list. Examples of categories are: Requirement specifications, executable code, Architecture, Design *etc*.

Sub Category: There may be many subcategories in present categories. Once the category is selected, then the subcategories present in these categories are also listed. If the component belongs to any of the present subcategories, then it is selected else the new subcategory is specified. Examples of subcategories are: Java, Ms.net, behavioral based patterns *etc*.

4.2.3 Steps in Repository Construction

- Defining the Purpose of the Repository: The present repository has been created for the purpose of efficient storage of software components so that they can be effectively retrieved.
- Defining the Services provided by the Repository: The repository provides with an organized view of the components present in it for retrieving them for reuse.
- Repository Software: The repository has been constructed using Microsoft SQL server 2005.
- Organization of the Repository: The repository has been organized to contain the various components divided into the respective categories and subcategories. The repository also contains the databases containing the surrogates of these components.

- Repository Management: The repository is constructed on the lines of data independence from the interface. The interface may keep on changing but the repository remains independent. Databases managing the various units such as the administrative unit, registration of users, various categories and sub categories have been also included. These are committed and updated according to the changes made with the provided authorization.

The database in the present system has been created with the name of "**work**". The main tables in the database are tablelogin (to maintain login information of the users.), admin (to maintain information about the administrator), comp(to maintain information about categories and subcategories), component(to maintain information about the various components in the database), nonsense(containing nonsense words).

Figure 4.1 shows, the database 'work' as seen in the Microsoft SQL Server Management Studio Express.

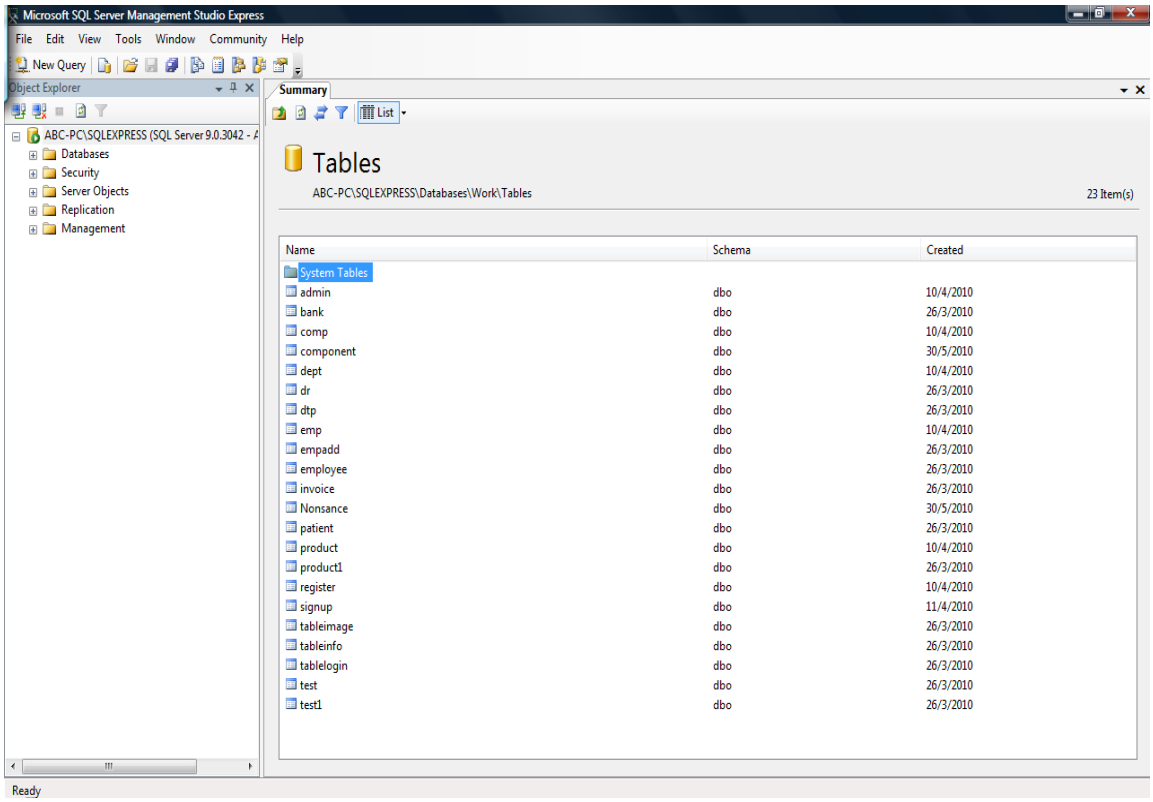

Figure 4.1: Snap Shot of the Database 'work'

4.3 Search and Retrieval of components

Once the repository has been created, then the next task is of the search and finally the retrieval of the desired components by the user. In order to restrict the search for the users, the components have been divided into seven basic categories of reuse which are:

- Executable code
- Source code
- Requirement specifications
- Test data
- Design
- Architecture
- Documentation

Further, more categories can be added by the users as per their requirements. Also there are some subcategories which further bifurcate the components present in the categories into more precise divisions. These subcategories can also be added by the users. The present system 'Effective Retrieval' implements three techniques of search and retrieval of components.

- Search and Retrieval based on Un - Indexed Components.
- Search and Retrieval based on Indexed Components
- Search and Retrieval based on Keywords.

4.3.1 Search and Retrieval from Un-Indexed List

If the user wishes to view all the components present in a particular category and browse through the entire list of the retrieved components, then this method has to be selected. Once the user is able to locate the asset of any relevancy, he can download the asset and use it accordingly.

The process however is very tedious. As the numbers of components in the repository keep on increasing, the browsing time also keeps on increasing. The possibility of locating and retrieving the required component keeps on slimming.

Also there can be instances of misunderstanding on the part of the user; the user might not understand the entire functionality of the component by just its name so he may has to browse various inappropriate components to find and retrieve the relevant one.

This type of process though very effective in small systems is not very successful in the systems with large number of components. So, it may be said that the efficiency of this system and the browse time are directly proportional to the number of components in the repository.

Un-Indexed Retrieval can be implemented in two ways:

- Displaying all the components in a category irrespective of the subcategories present.
- Displaying the components of a selected sub category of a particular category.

The 'Effective Retrieval System' displays the components present in a particular sub category of a category.

The Figure 4.2, shows a flowchart of search and retrieval based on the un- indexed list. The user is required to select a category from the list of categories present. After selecting the category, the user shall select any subcategory if it is present. Once the selection has been made, then the user is able to view the list of the assets which are retrieved on these two bases. The user then browses through the list of the displayed assets and find the one most relevant for his purpose. The user has to browse through the entire list of the displayed assets manually, to find the asset required.

At the backend, when the category and subcategory is selected, then the database containing the information about the components contained in various categories and subcategories is visited and the ones in the present context are chosen and displayed. The database is created by extracting information from the surrogates of the various components.

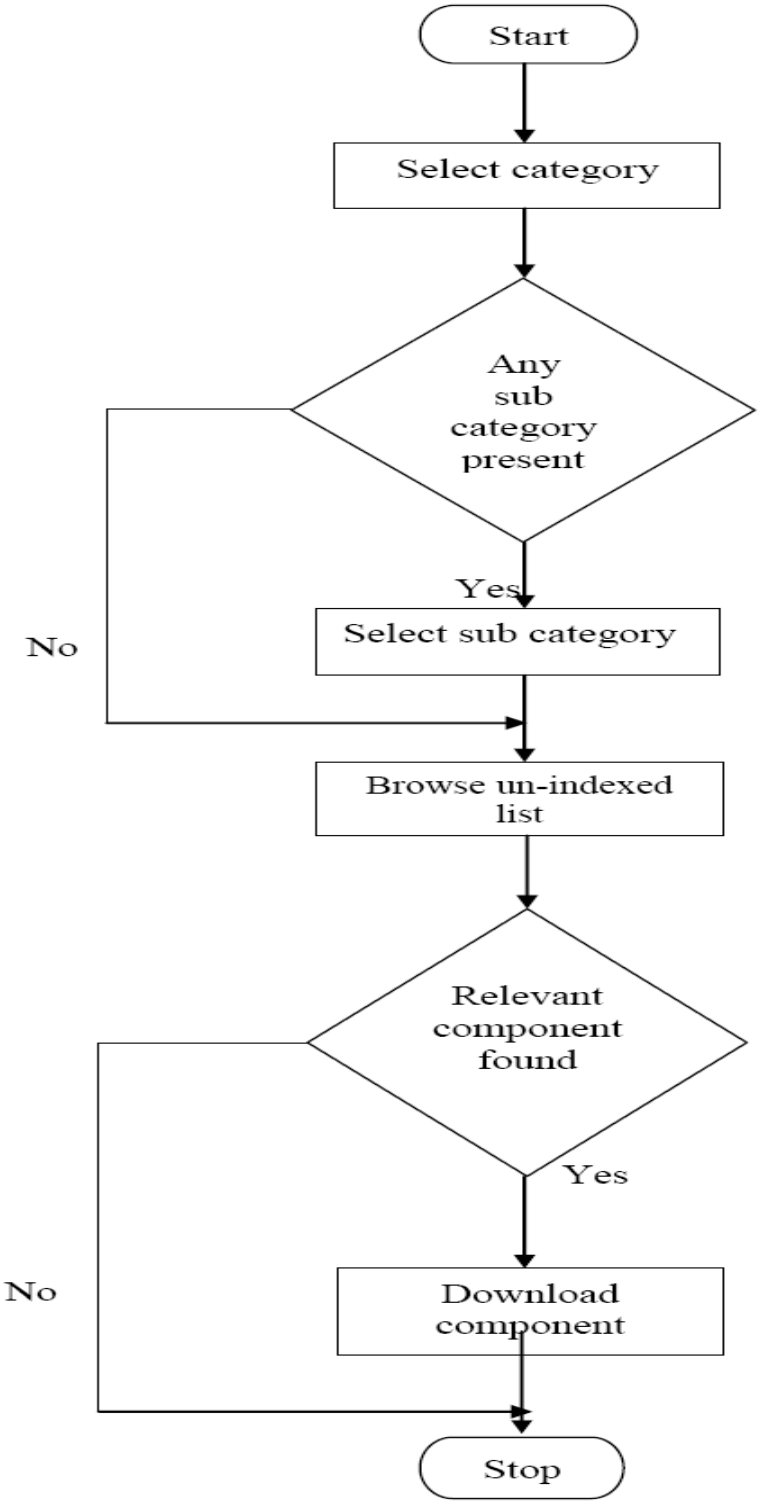

Figure 4.2: Search and Retrieval from Un-Indexed list

4.3.2 Search and Retrieval from Indexed List

Indexed list represents the components indexed by enumerated classification. Enumerated classification is a well-known retrieval method used by the Dewey Decimal system and ACM's *Computing Reviews* Classification System. In this method, information is placed in categories that are usually structured in a hierarchy of sub categories; the appeal of the classification scheme is the ability to iteratively divide an information space into smaller pieces that reduces the amount of information that needs to be perused [8].

The Indexes can be created on the basis of:

- Alphabetically: The components in this list are arranged in the alphabetical order. As soon as the component is added, it is enumerated in the list to be displayed. This type of list is helpful in case the user knows exactly the name of the component he wants to retrieve. The user can browse down exactly to the place containing components starting from the first alphabet of the component required.
- According to relevancy: This type of list enumerates the components according to the most relevant to least relevant. The relevancy of a component is directly proportional to the number of times it is downloaded.
- According to version: The list enumerates the components according to the latest version to the oldest version. The user can then choose from the list according to his software and hardware specifications.

The 'Effective Retrieval System' implements generates indexes enumerated both alphabetically and according to version.

Figure 4.3, shows the flowchart of the search and retrieval of components using the indexed list. The user wishing to view the components in the form of an enumerated list shall select the relevant category and subcategory (if present), then will browse through the displayed tree view of the retrieved components to select the relevant asset and download it.

At the backend PEEL (**P**arse and **E**xtract **E**macs **L**isp) is used to extract text related to categories and subcategories from the surrogate and form enumerated indexes.

The drawback of using this approach is that, as the numbers of components keep on increasing, the list keeps on becoming more and more tedious to view also there is inherent inflexibility in the lis

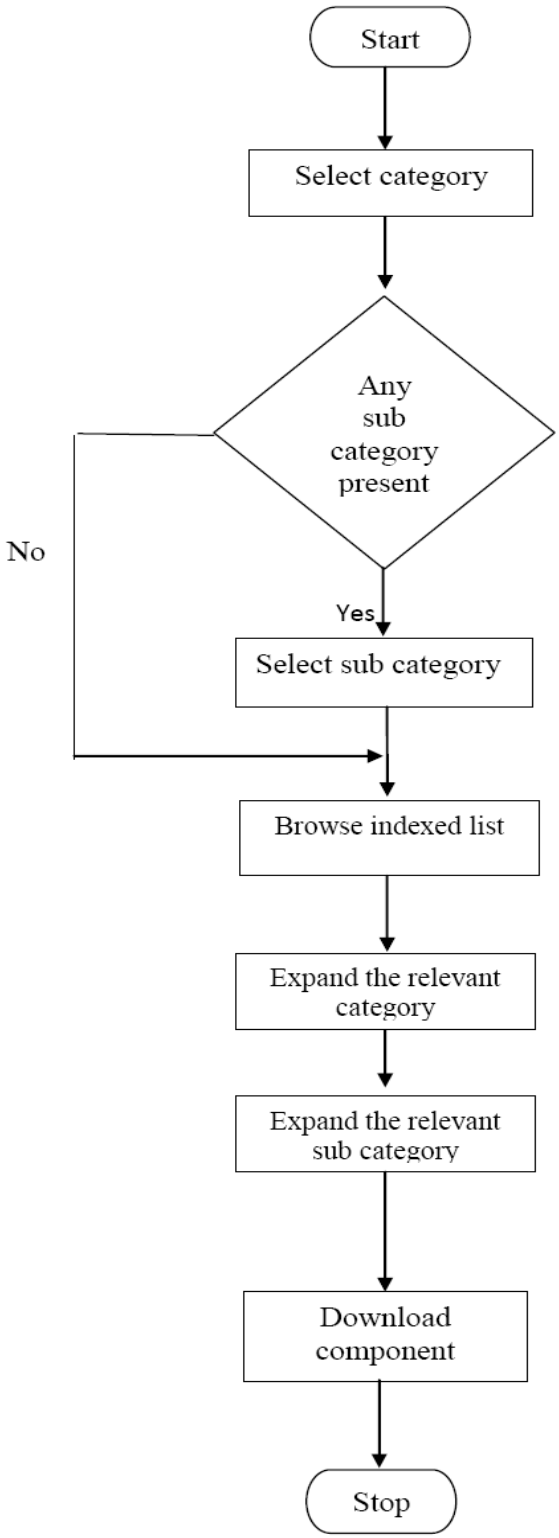

Figure 4.3: Search and Retrieval from Indexed List

4.3.3 Search and Retrieval Based on Specific Criteria

The user can also search the repository by typing his requirement in a search engine and then further selecting the criteria on which he wants his search to be based in order to restrict the search and make it more precise. The various criteria are:

- **Text Based Search:** In this case, the entire surrogate is searched and the components found valid according the given search are displayed.
- **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 the related components.
- **Signature Based Search:** 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.
- **Input/ Output Based Search:** In this case the user shall provide with the input to be given by him to the system and the expected output from the system. Here only the input/output part of the surrogate is searched and the relevant result is displayed.

Also, in order to compare the various search and retrieval techniques, automation of finding out precision, recall and coverage ratio has been done. This has been implemented by setting up a clear counter, every time the results have to be taken for any of the techniques, clear counter is pressed which shall reset the counter and start keeping count of the relevant assets by incrementing each time a component is downloaded by the user.

Figure 4.4, shows search and retrieval based on specific criteria. The user is required to input the text for searching and select the search criteria. After doing so, the relevant retrieved assets are displayed. Once the user finds the most appropriate component, he can download it and if it is not found then the user can again perform the search.

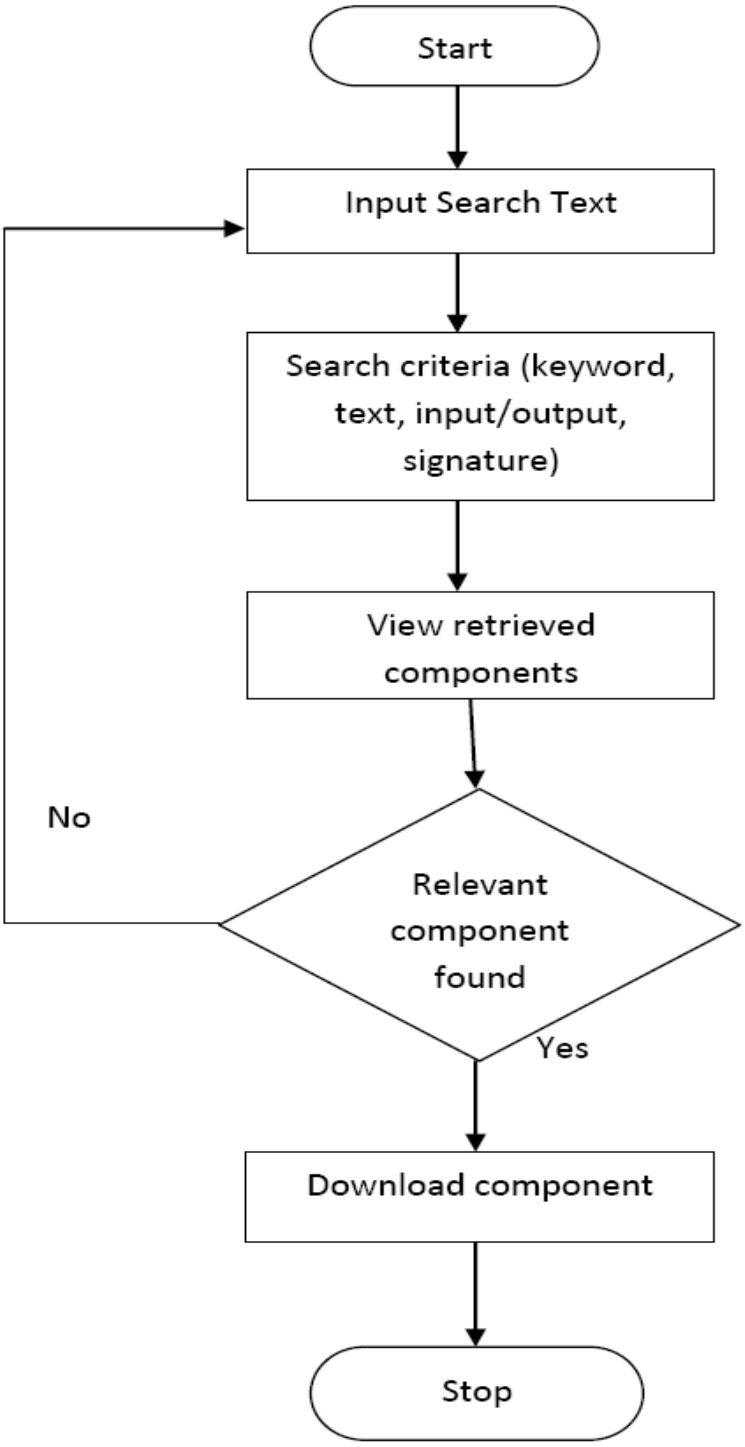

Figure 4.4: Search and Retrieval Based on Specific Criteria

At the backend, PEEL (**P**arse and **E**xtract **E**macs **L**isp) system is implemented in order to extract information from the surrogate and index it for the purpose of performing effective search.

Firstly, the entire surrogate is scanned and the nonsense words such as “and,” “of,” “the,” and others are removed and the keywords, signatures, input/output are extracted and linked with the components. Once the user writes the text in the search engine and selects the criteria of search, the matching components are found and displayed for the user to retrieve and download.

Figure 4.5, shows the snap shot of the database containing the list of nonsense words with which the entire text is compared to extract important information.

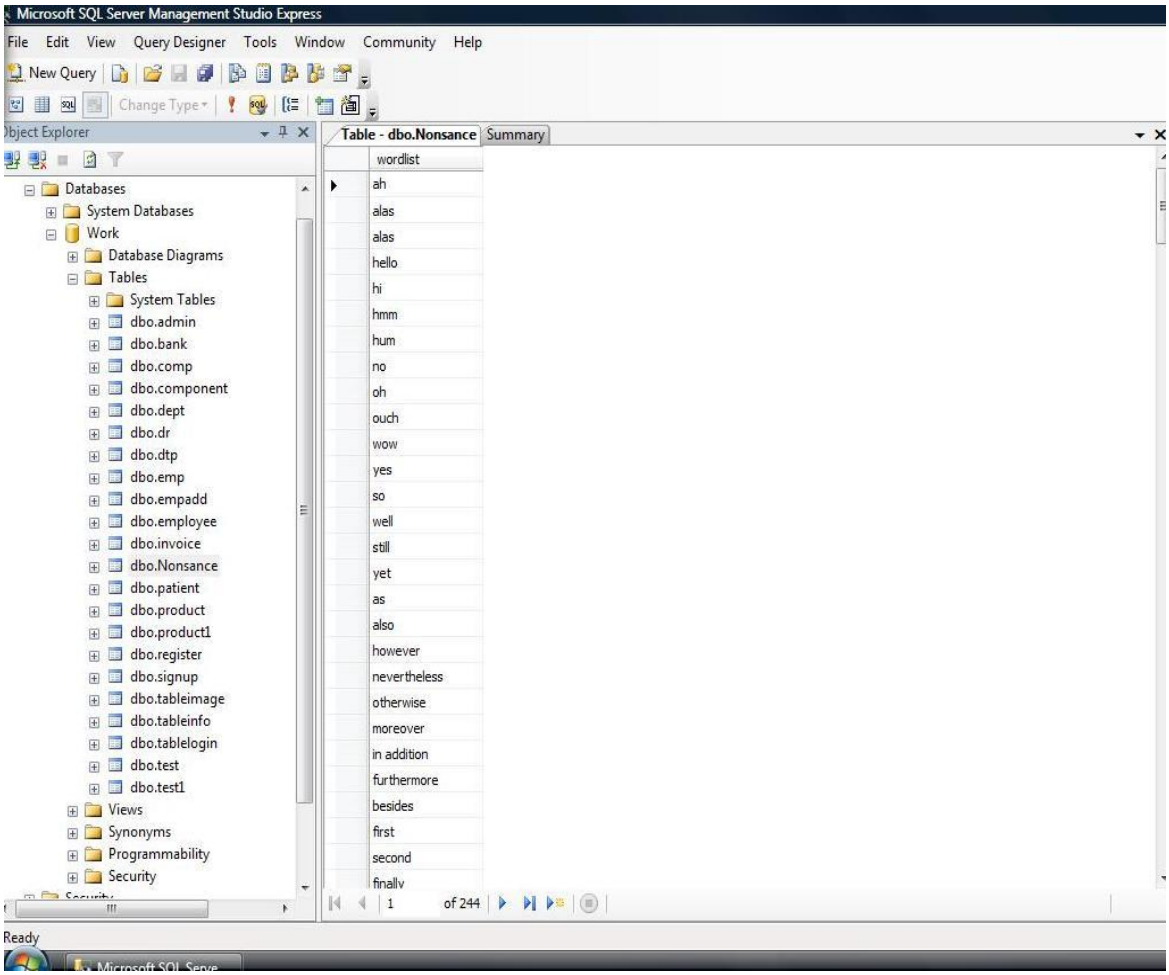

Figure 4.5: Snap Shot of the ‘Nonsense’ Table

4.4 Working of Effective Retrieval System

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

- **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. The unregistered users can only view these components and not download them.

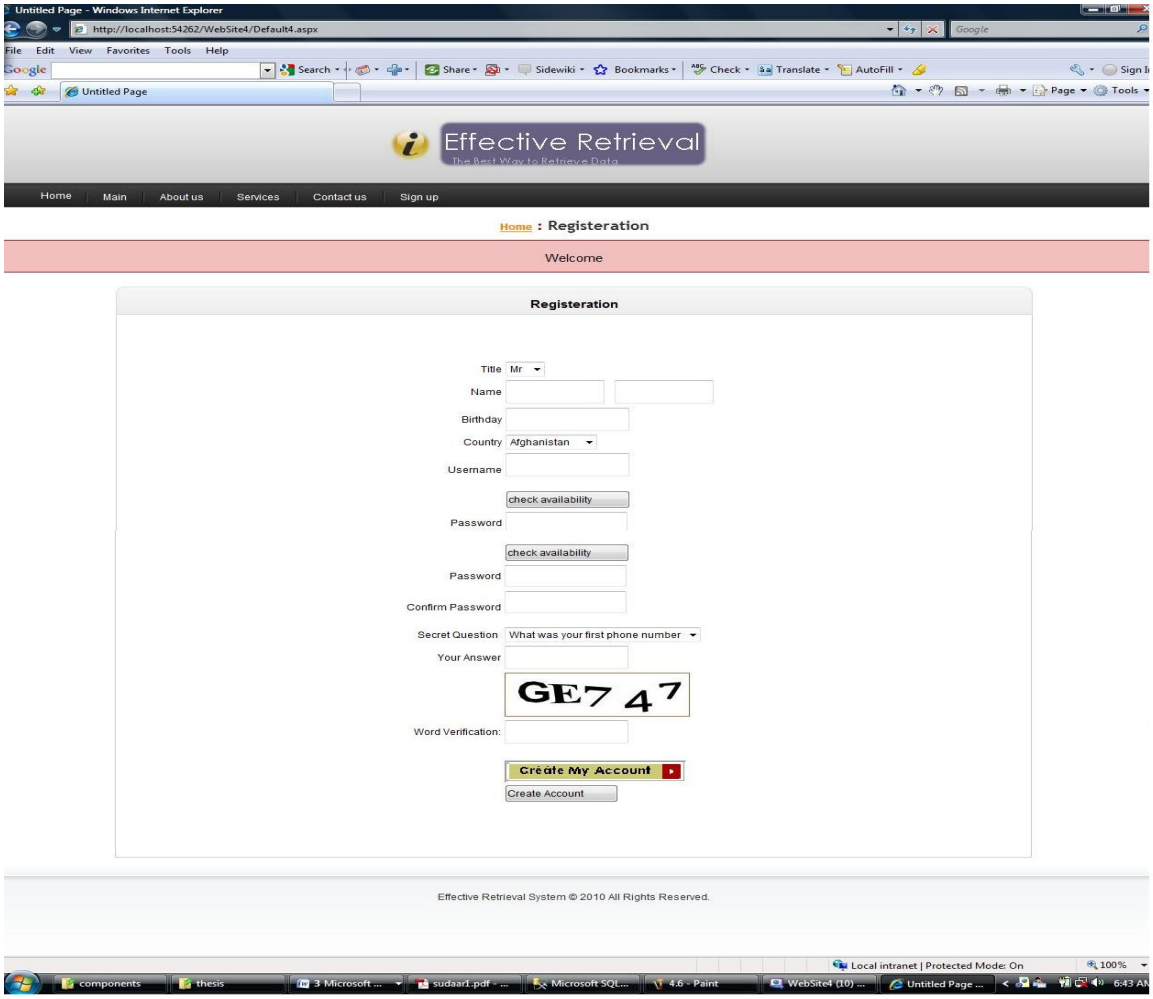

Figure 4.6: Snap Shot of User Registration Form

- **User Login**

The registered users have 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 4.7, shows the User Login area where the user has to enter his authenticated user name and password to access the system.

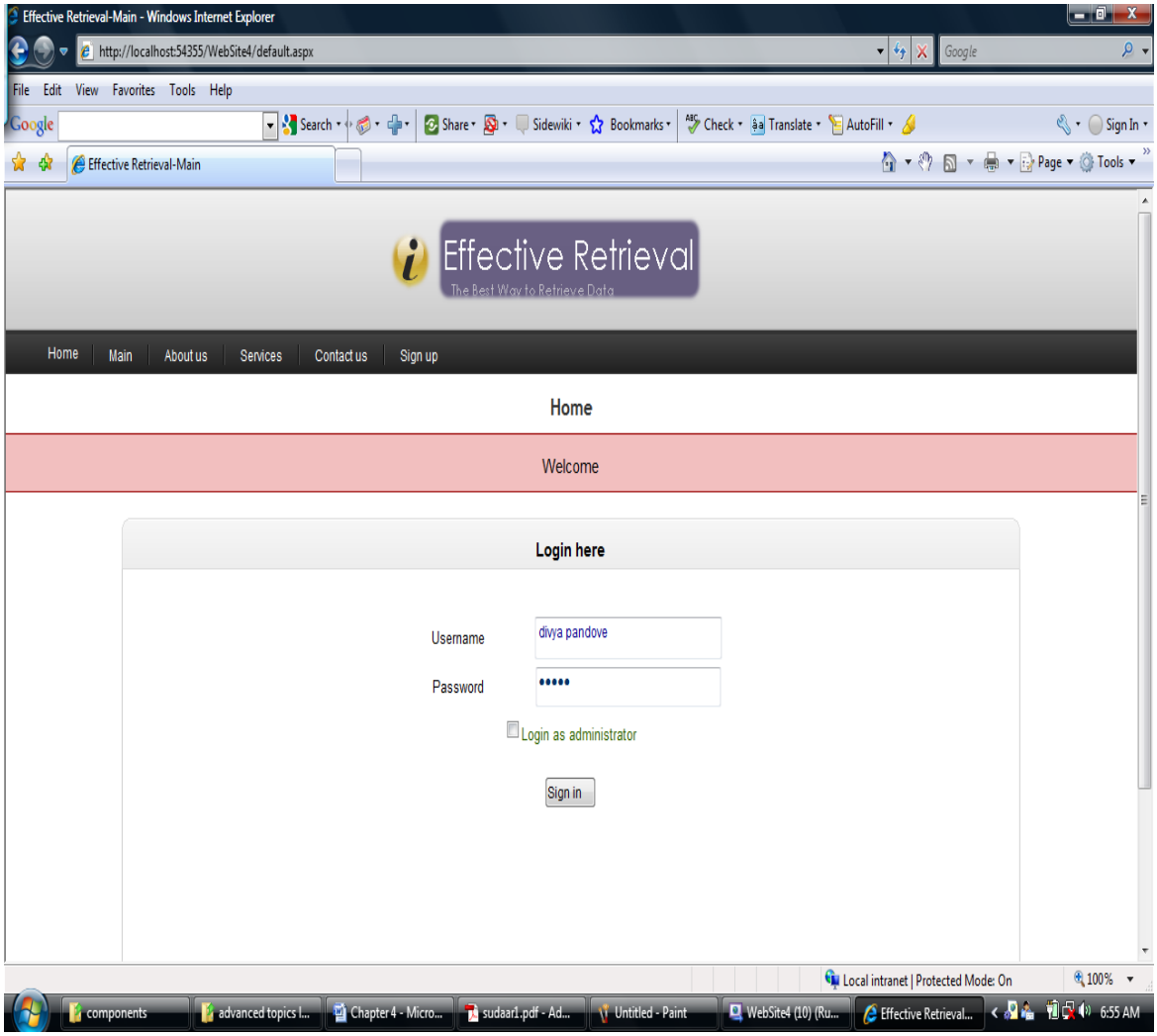

Figure 4.7: Snap Shot Of User Login Area

- **Administrative Control**

The administrator controls and updates the entire system. The administrator can perform following functions:

- Add New Category: Addition of any new category in the list of the basic categories.
- Add New Sub Category: Addition of any new subcategory under the present categories.
- Delete/Update Components: Administrator can delete any present component or can also update the existing component.
- View User Added Categories: Administrator can view any of the categories that the user adds and then decide to include or not include it in the basic categories.
- View User Added Subcategory: This facilitates administrator to add any subcategory to a given category according to his prerogative.

Figures 4.8, 4.9, 4.10 and 4.11, show the Administrative Control area as present in the Effective Retrieval System.

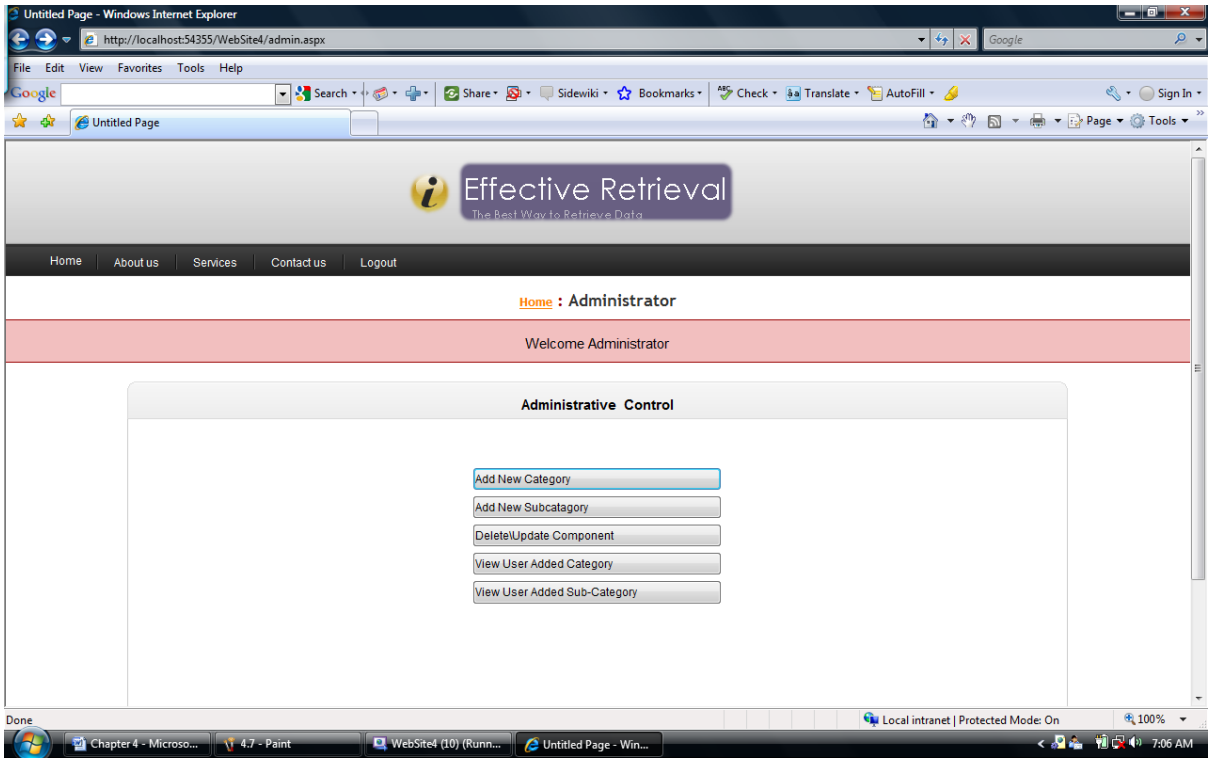

Figure 4.8: Snap Shot of Administrative Control

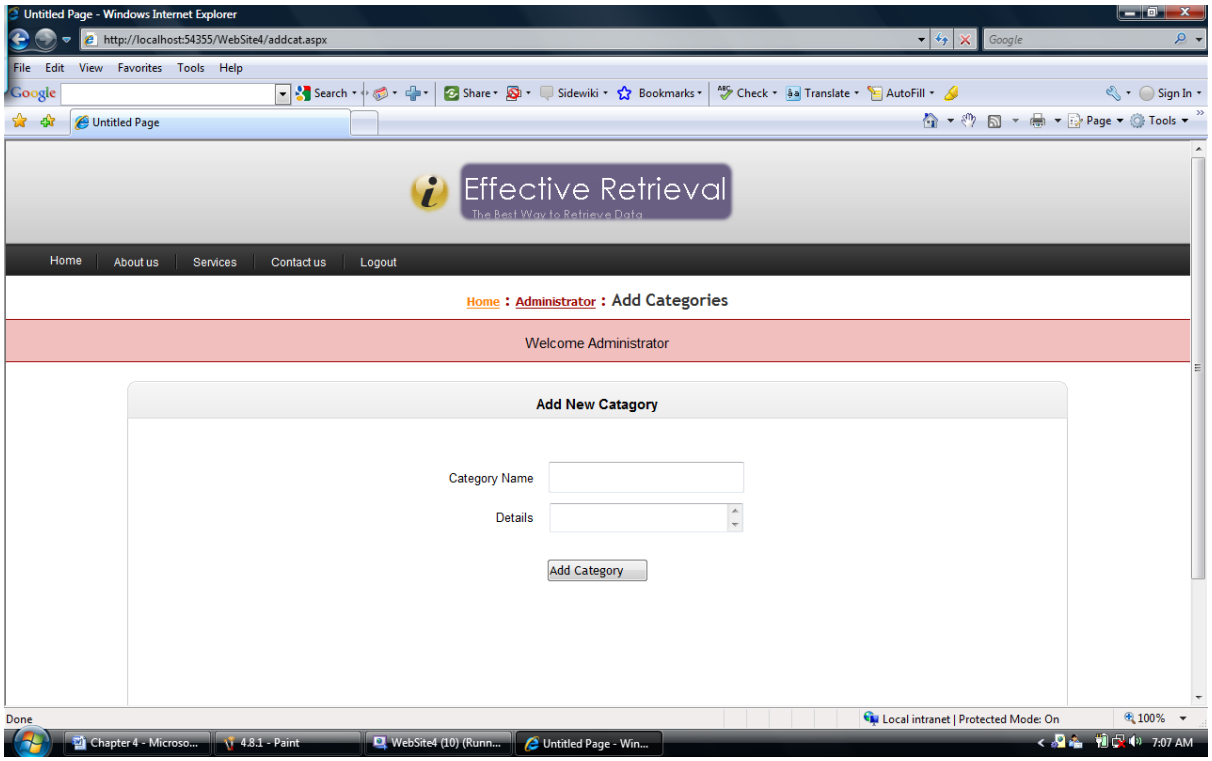

Figure 4.9: Snap Shot of Add New Category

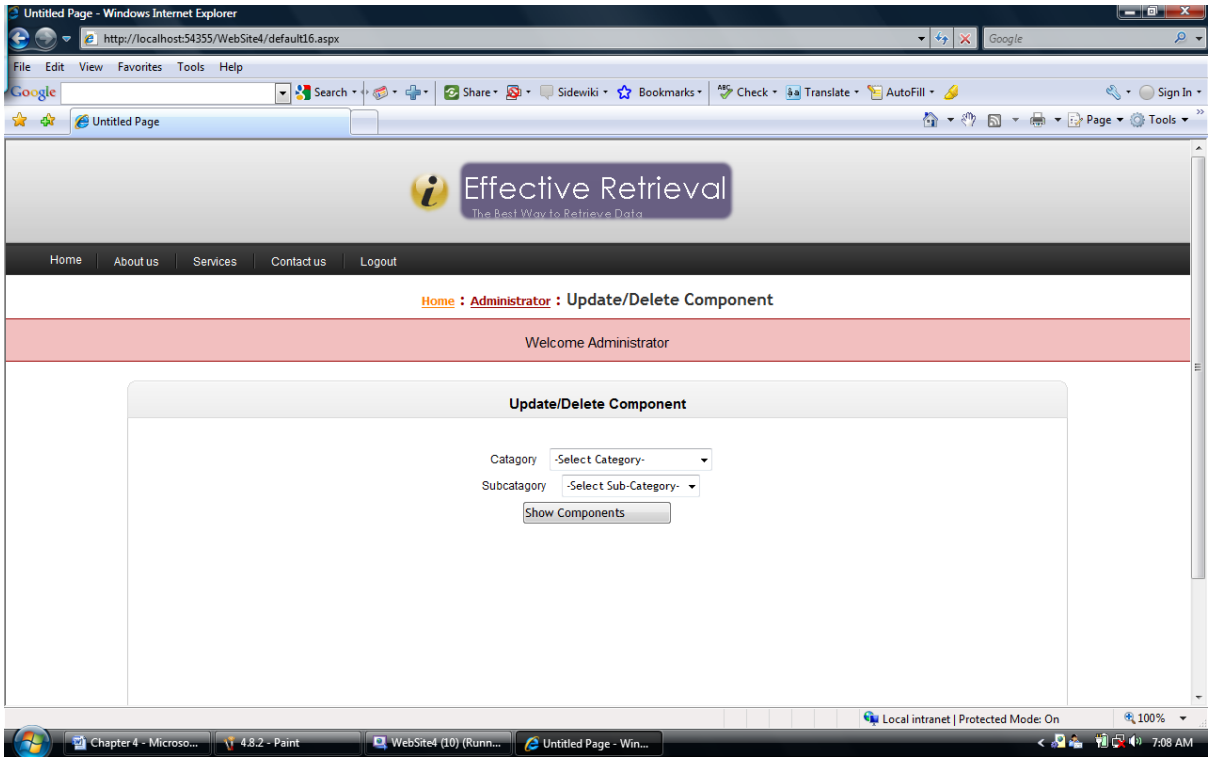

Figure 4.10: Snap Shot of Update/Delete Components

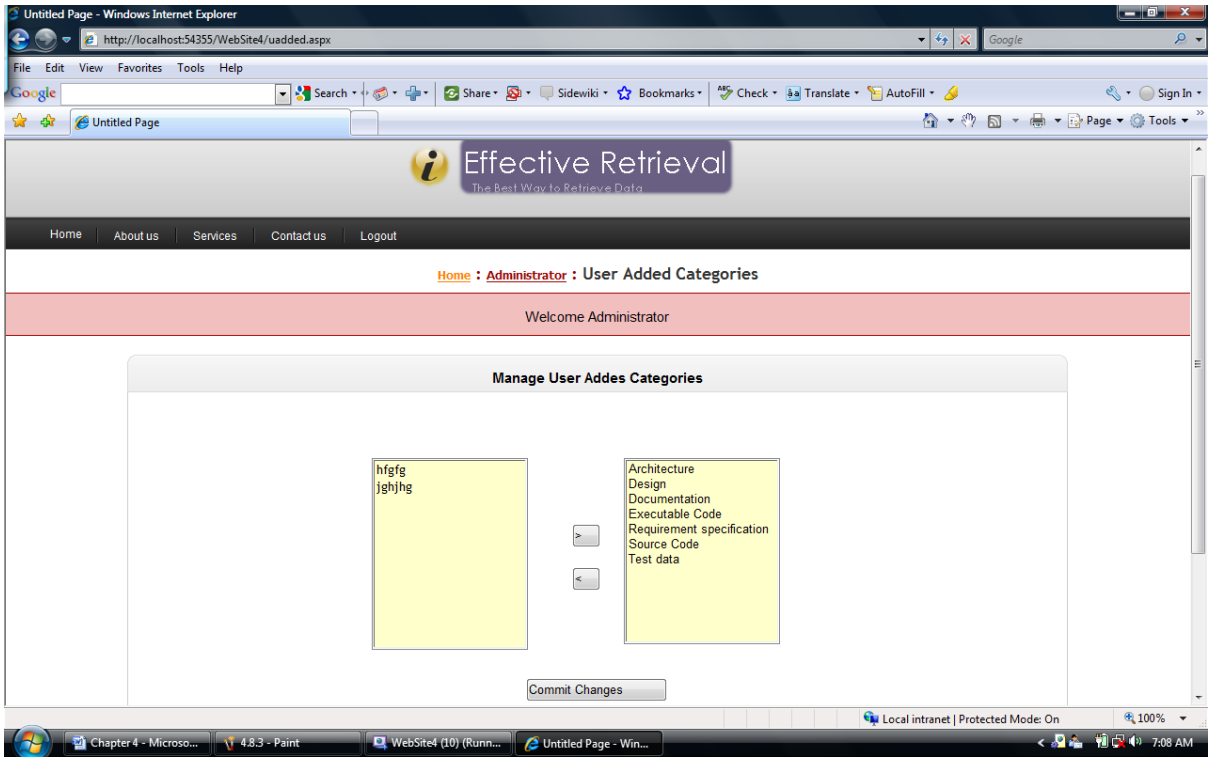

Figure 4.11: Snap Shot of User Added Categories

- **Main Home Page**

The main homepage is divided into three broad areas:

- Category Based Search: Here the user types his requirements in the search engine and selects the category for the purpose of specific search.
- Un-Indexed Retrieval: Here the user needs to select the category and subcategory of the component required and view all the components present in the selected context.
- Indexed Retrieval: Here an enumerated list of the selected category and subcategory is displayed from where the used can select the desired component.

Figure 4.12, shows the preview of the main home page of the system.

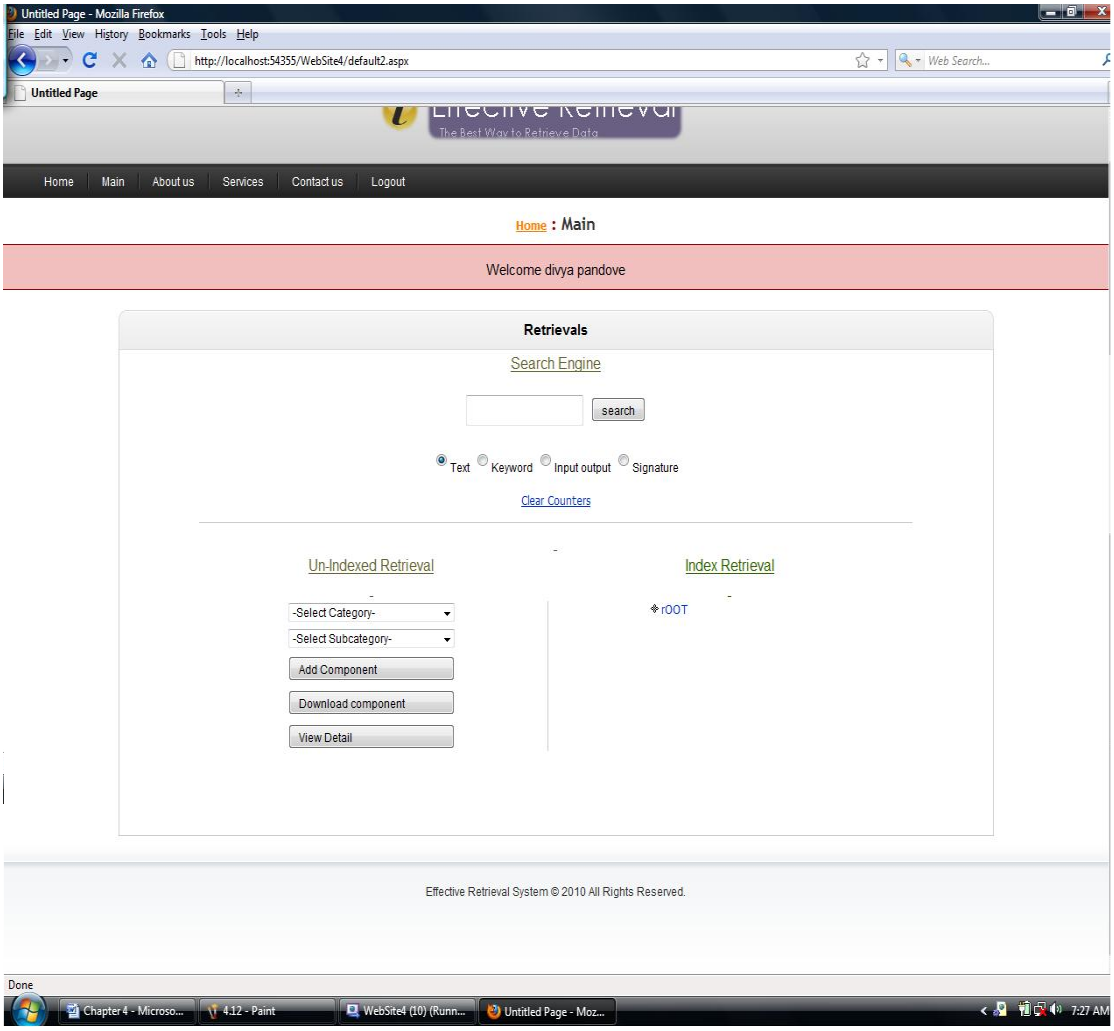

Figure 4.12: Snap Shot of the Main Home Page

- **Add Component**

This page helps to prepare the surrogate as before uploading any component the user needs to add the information regarding the present component. Figure 4.13, shows the preview of the area that takes as input the surrogate of the components and also helps in uploading the component.

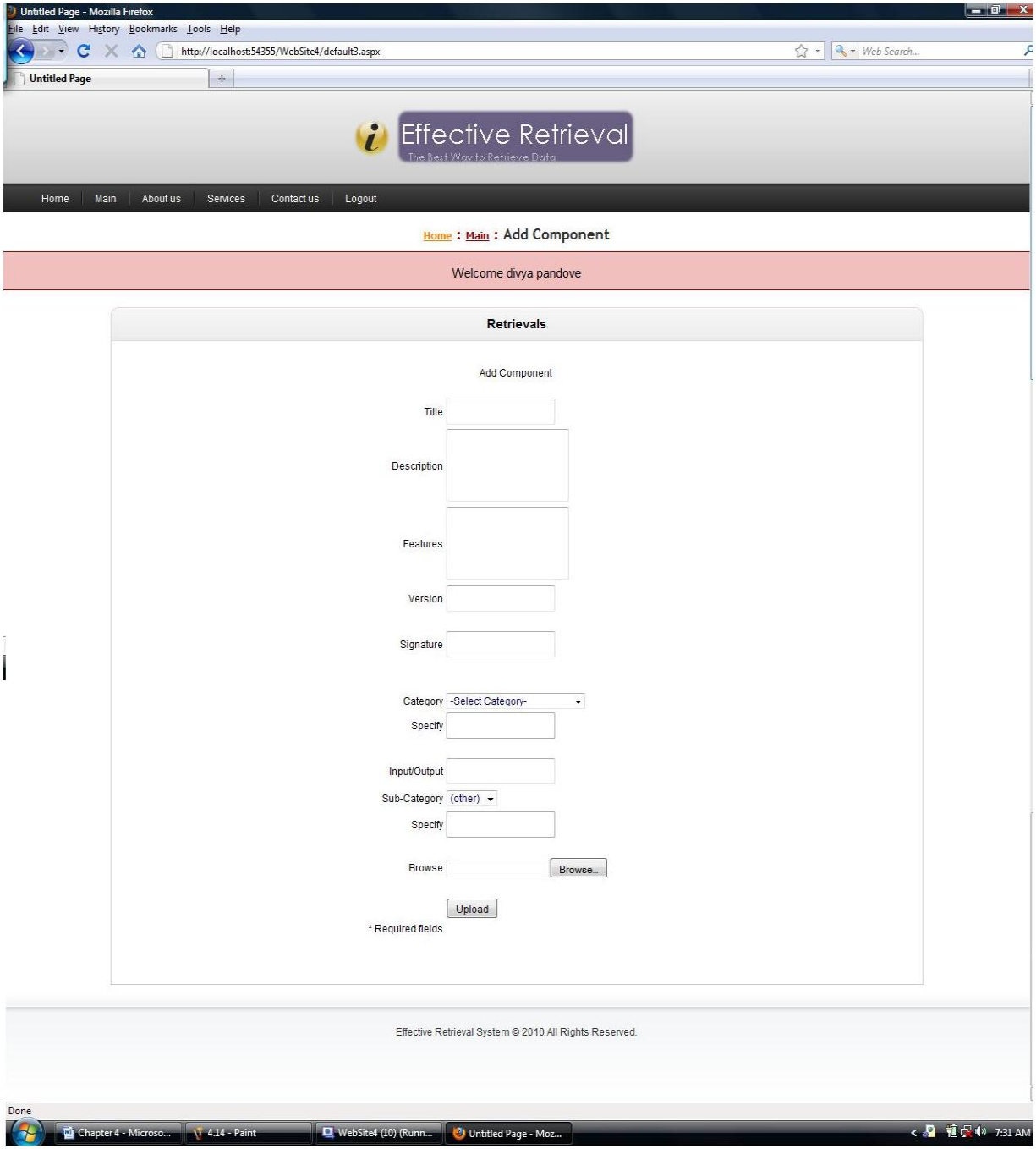

Figure 4.13: Snap Shot of Add component.

Chapter 5

Testing of the Developed System

5.1 Introduction to Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test [31].

Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, 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.

5.2 Types of Testing Employed on the System

The system has undergone many levels and kinds of testing methodologies; this section defines these methodologies and explains how the system has been tested with them.

5.2.1 Unit Testing

Unit Testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors [32].

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

- Administrative unit.
- Indexed Retrieval unit.

- Un-Indexed Retrieval unit.
- Search Engine based unit.

5.2.2 Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system [33].

The entire system after being integrated into one website has also been tested as a whole 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 tests a completely integrated system to verify that it meets its requirements [34]. 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 Testing for Data Integration

Data Integration involves combining data residing in different sources and providing users with a unified view of these data [35]. The system has been tested to see if it implements data integration or not. This has been done in order to make sure that the interface is independent of the database. In order to do so, ten versions of the system have been deployed and the result was that in nine cases the database was not required to be altered. Only once when actually a new field was added, the database was needed to be

modified in order to provide a storage structure for the elements of that field. Figure 5.1, shows a simple schema implementing the data integration.

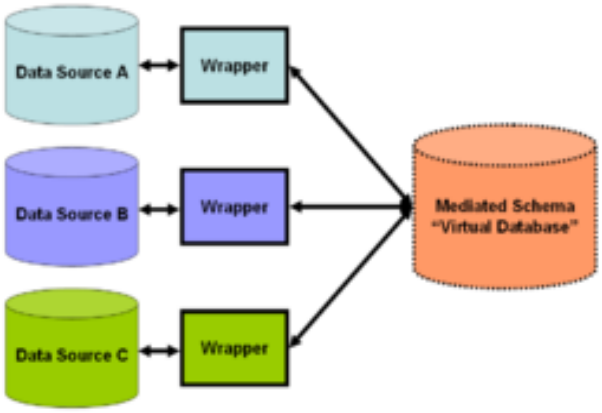

Figure 5.1: Simple Schema for Data Integration [36]

5.2.5 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 10 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.

5.2.6 Beta Testing

Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users [36].

The system has been hosted on the internet in order for it to undergo beta testing. The URL for the same is:

www.aspspider.ws/effectiveretrieval.com

5.3 Results Obtained After Testing the System

Some results on the basis of the three parameters of precision, recall and coverage 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:

- Indexed and Un-indexed retrieval.
- Keyword based, Input/output based, signature based and text based searches.

5.3.1 Results of Comparison between Un-Indexed and Indexed Retrieval

A sample size of 5 components has been taken and results have been compared on the basis of precision, recall and coverage ratio, automatically generated by the system. The 5 sample components have been randomly selected from following categories and subcategories.

Component number	Category	Sub category
1	Architecture	---------
2	Design	Behavioural patterns
3	Executable Code	Ms.net
4	Documentation	---------
5	Requirement specifications	---------

Table 5.1: Component Selection

5.3.1.1 Results Obtained for Un-Indexed Retrieval

Components / Comp. criteria	1	2	3	4	5
Precision	0.2	0.0769	0.166	0.2	0.5
Recall	0.5	0.166	0.66	0.4	0.166
C. Ratio	0.0625	0.325	0.15	0.125	0.075

Table 5.2: Un-Indexed Retrieval results

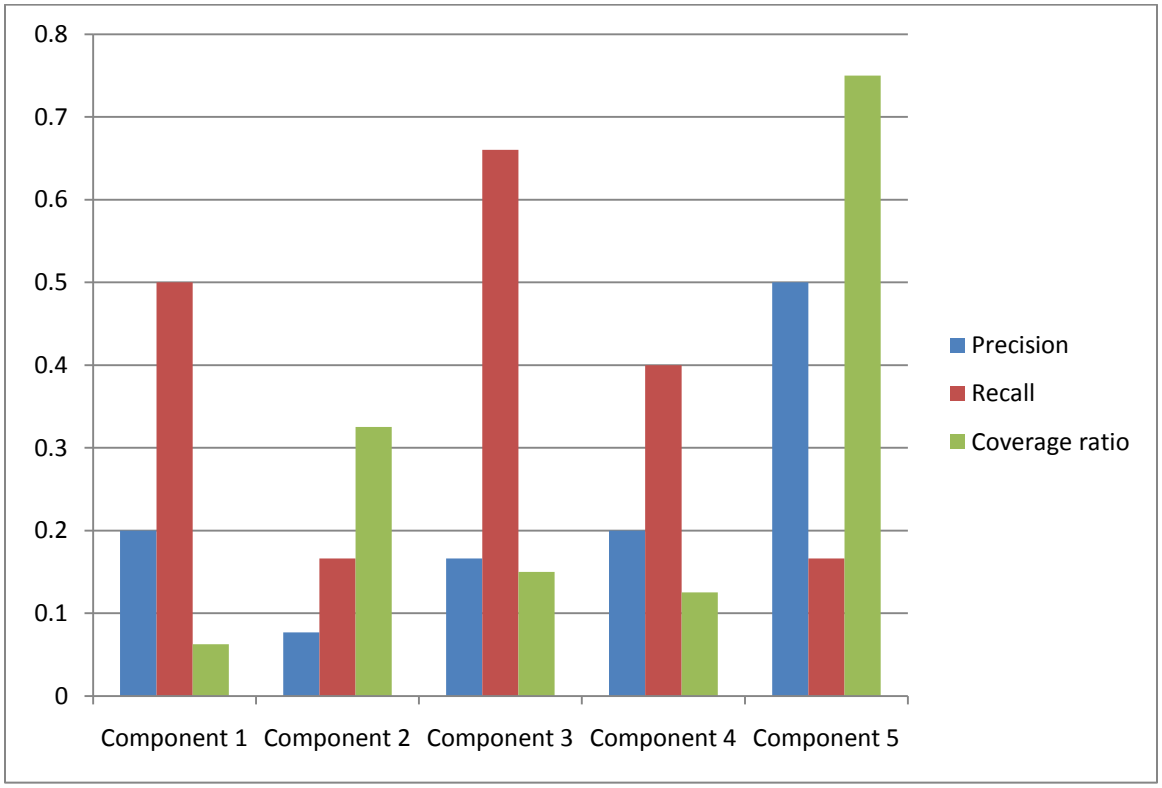

Figure 5.2: Graph Showing Un-Indexed Retrieval Results

5.3.1.2 Results Obtained for Indexed Retrieval

Components / Comp. criteria	1	2	3	4	5
Precision	0.2	0.33	0.28	0.2	0.5
Recall	0.5	0.166	0.66	0.4	0.166
C. Ratio	0.0625	0.15	0.0875	0.125	0.075

Table 5.3: Indexed Retrieval Results

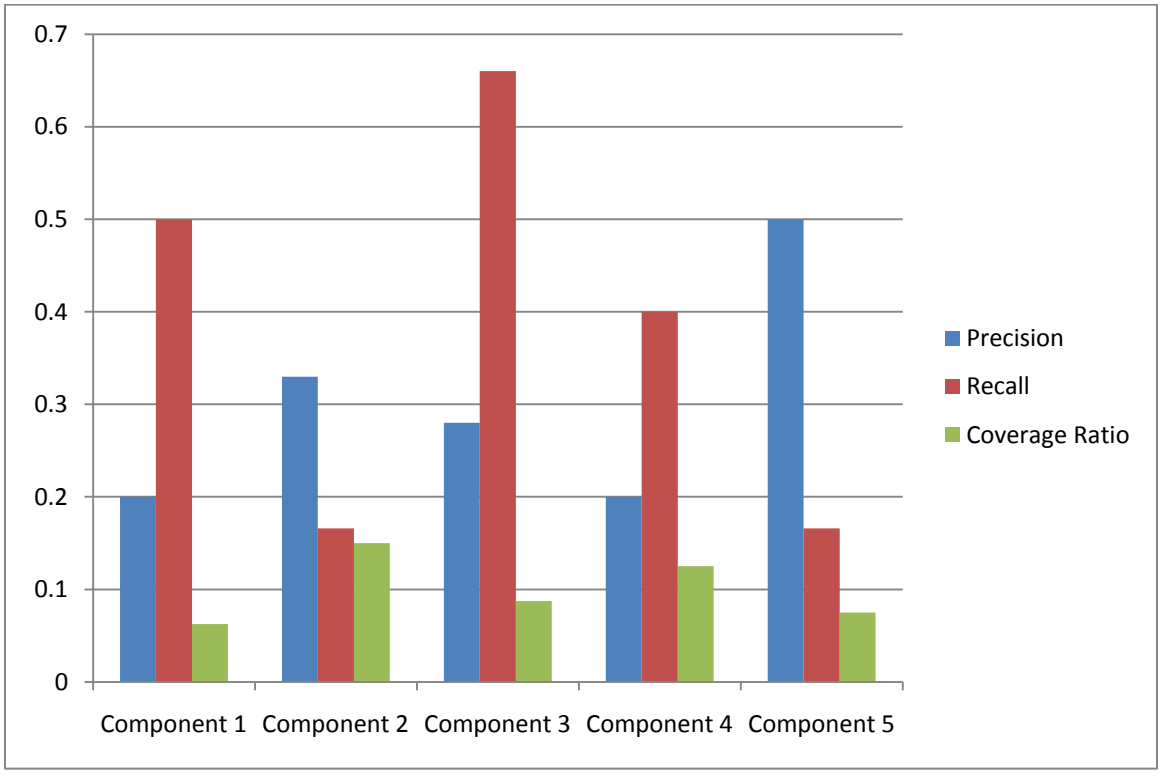

Figure 5.3: Graph Showing Indexed Retrieval Results

The comparison between the two retrieval techniques show that the values obtained for precision, recall and coverage ratio of the indexed components are more encouraging than the ones obtained for un-indexed components.

For instance if we compare the values obtained for the third component, we find that the precision of Un-Indexed retrieval *i.e.* 0.16 is less than that of the Indexed one *i.e.*0.28, so it can be concluded that the components obtained from the Indexed list are more close to the user requirements.

Similarly, on comparing the values of recall we find that the value of Indexed retrieval and Un-Indexed retrieval is the same *i.e.*0.66 which proves that both the techniques, approximately, give back the same number of total relevant assets. Now the coverage ratio of the Indexed retrieval *i.e.*0.0878 is much less than that of the Un-Indexed retrieval *i.e.* 0.15, the main reason behind this is that in the case of Un-Indexed retrieval, all the components in a category and it's various subcategories are visited but in case of the Indexed retrieval only the components of the subcategory that has been expanded will be visited.

Hence, it can be concluded that Indexed retrieval is better than un-indexed retrieval when it comes to selecting the components from the subcategories of the categories present. As due to faceted enumeration, the components become easier to retrieve.

5.3.2 Results of Comparison between Various Search Engine Criteria

The input given to the search engine is on the basis of the various search engine criterion (keyword, input/output , signature, text) is given along with keeping in mind the retrieval of the components sample as chosen earlier.

5.3.2.1 Results Obtained for Text Based Search

Components / Comp. criteria	1	2	3	4	5
Precision	0.4	0.22	0.28	0.4	0.33
Recall	0.66	0.33	0.66	0.66	0.5
C. Ratio	0.0625	0.325	0.0875	0.125	0.075

Table 5.4: Text Based Search Results

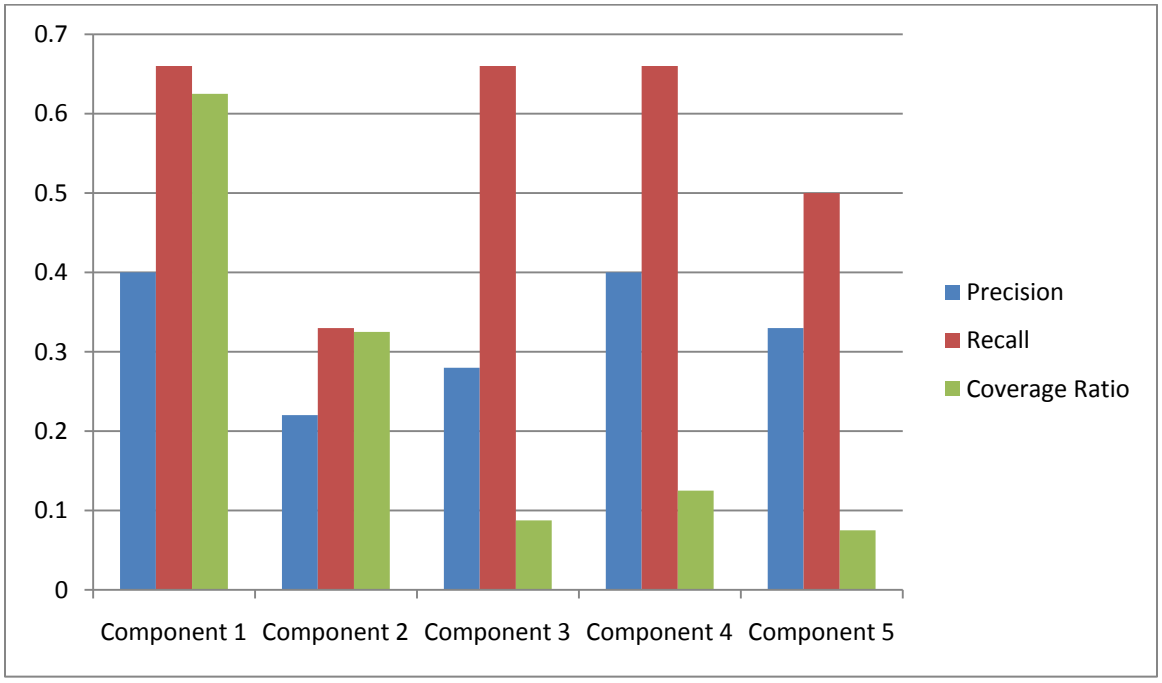

Figure 5.4: Graph Showing Text Based Search Results

5.3.2.2 Results Obtained for Input/ Output Based Search

Components / Comp. Criteria	1	2	3	4	5
Precision	0.5	0.22	0.5	0.5	0.33
Recall	0.5	0.25	0.66	0.33	0.5
C. Ratio	0.0625	0.325	0.0875	0.125	0.075

Table 5.5: Input/ Output Based Search Results

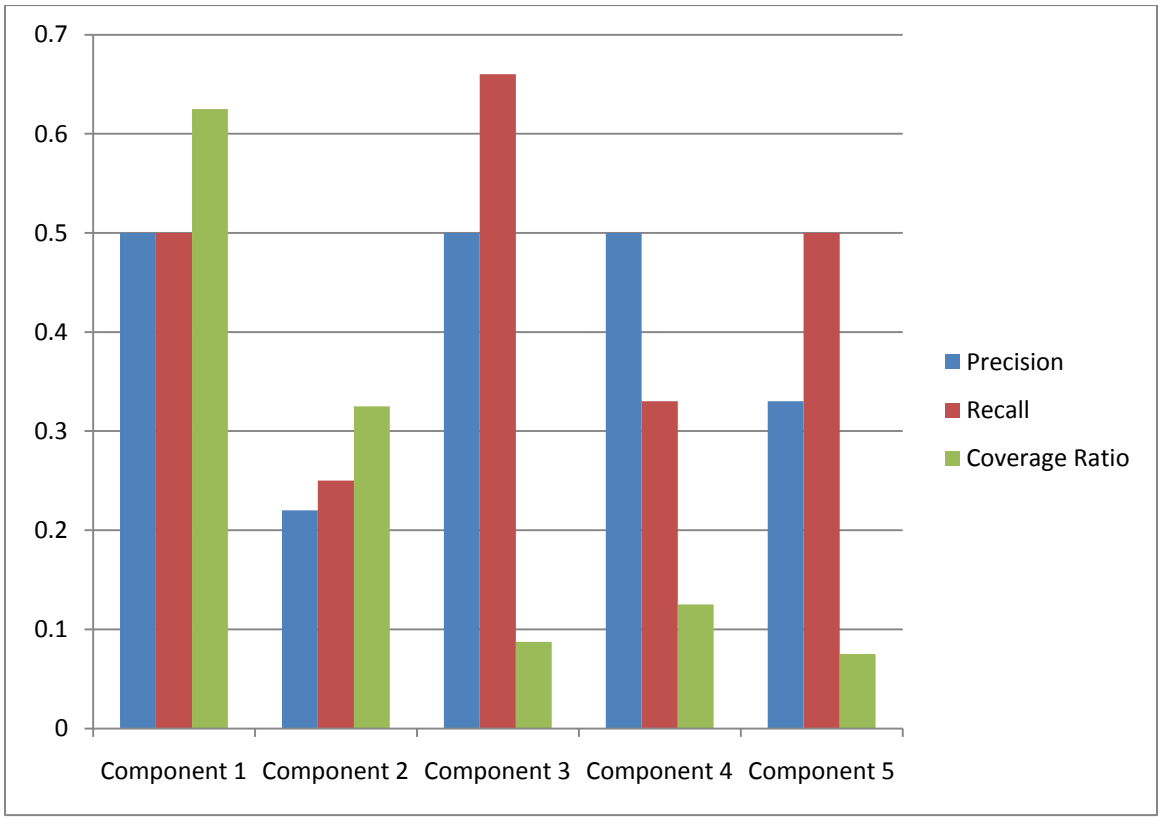

Figure 5.5: Graph Showing Input/ Output Based Search Results

5.3.2.3 Results Obtained for Signature Based Search

Components / Comp. criteria	1	2	3	4	5
Precision	0	0	0.5	0	0
Recall	0	0	1	0	0
C. Ratio	0	0	0.05	0	0

Table 5.6: Signature Based Search Results

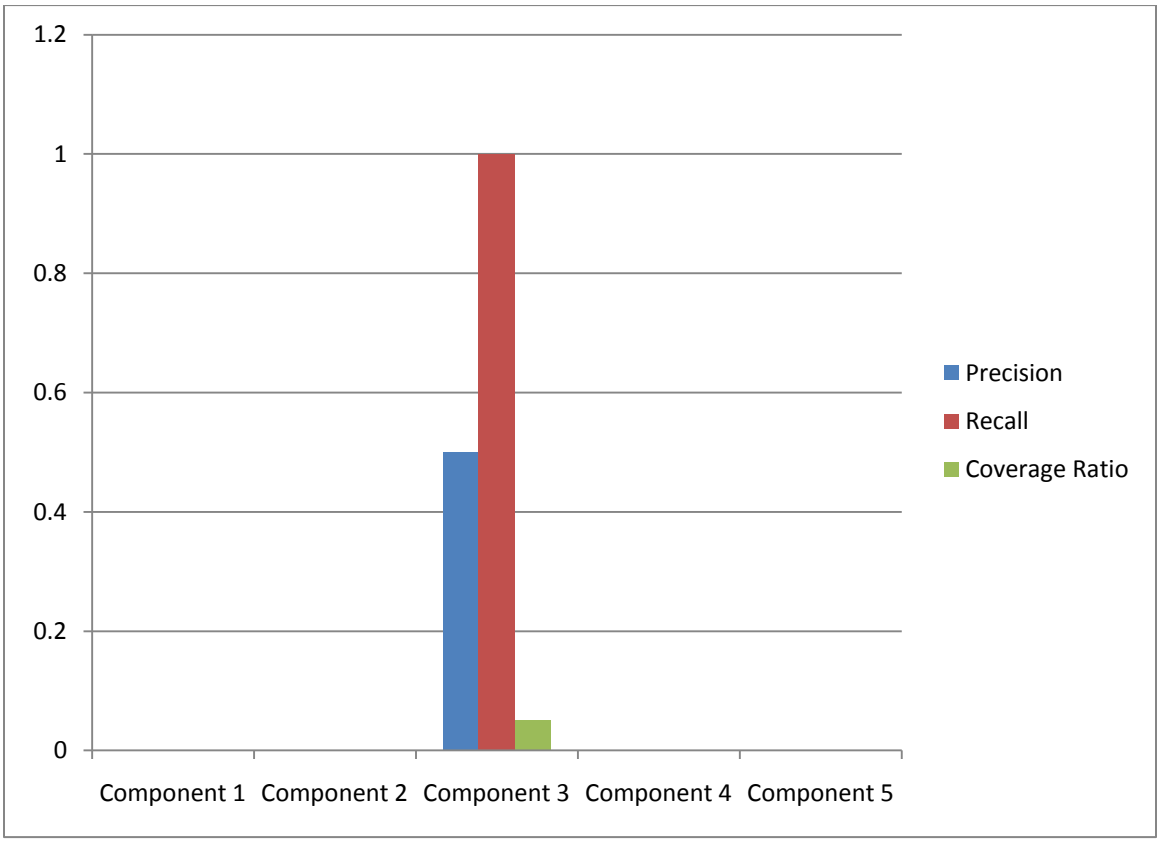

Figure 5.6: Graph Showing Signature Based Search Results

5.3.2.4 Results Obtained for Keyword Based Search

Components / Comp. criteria	1	2	3	4	5
Precision	0.5	0.33	0.142	0.5	0.5
Recall	0.5	0.66	0.33	0.83	0.75
C. Ratio	0.0625	0.15	0.0875	0.125	0.075

Table 5.7: Keyword Based Search Results

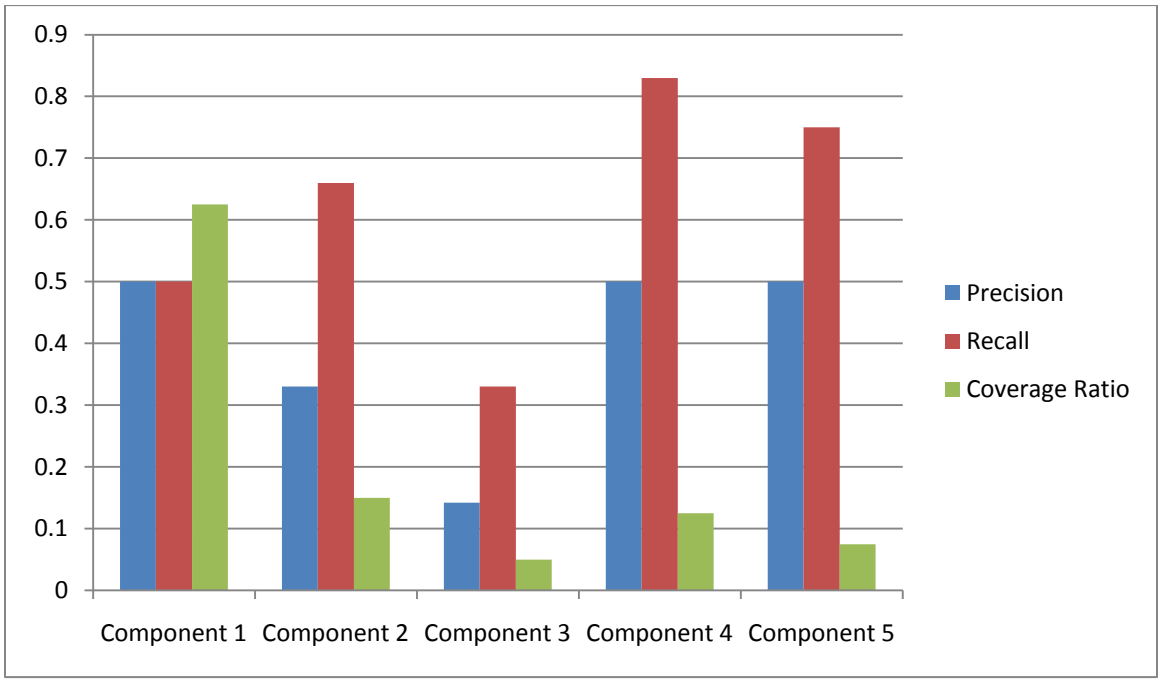

Figure 5.7: Graph Showing Keyword Based Search Results

Comparing the values of the precision, recall and coverage ratio for various techniques we find that the keyword based search show most encouraging results.

For instance, comparing the results of the third component we find out that precision of Input/ Output based search *i.e.* 0.5 and that of signature based search *i.e.* 0.5 may be more than keyword based search *i.e.* 0.142 and text based search *i.e.* 0.28 but this is due to the

fact that number of components having signature and input/output information stored in their surrogates is very less. Now, in the sample taken for testing four out of five components do not have any signature, so the value of precision in that case remains 0. Now, further when we compare the values of the recall criteria then we find that the values of the various techniques are:

Input/ Output based search = 0.33, Signature based search = 1, Keyword based search = 0.33, Text based search = 0.66.

Now, here the best recall seems that of Signature based search, the value '1' depicts perfect recall but again the number of components in the repository are very less and the search inputted is able to retrieve all the relevant components present.
Similarly, comparing the values for coverage ratio which are as follows:
Input / Output based = 0.125, Signature based = 0.05, Keyword based = 0.875, Text based = 0.875. Here, the keyword and text based searches give the maximum value as they cover the greatest area in the repository as opposed to the Signature based and Input/ Output based search. But, overall when we compare the values of all the components taken we find that the values of the Keyword based search are the most consistent.

Hence, it can be concluded that the keyword based search is the best amongst all the search criteria set up in the search engine.

Chapter 6

Conclusion and Future Scope

- **Conclusion**

The 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 and subcategories from where the user can extract the desired component effectively and efficiently. After implementation and testing of the system, the following conclusions come to light:

- Indexed retrieval method is preferred over the un-indexed retrieval method as the browsing time is reduced and the retrieved assets are more precise.
- The implementation of PEEL makes the search more precise and by changing the surrogate dynamically for each and every criterion in the criteria based search and the retrieval time is reduced and the search results are more exact.
- Out of all the search criteria the keyword based search is better as the user does not require memorizing the exact input, also in terms of the comparison criteria, it proves to be better than the rest of the criteria.

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:

- The entire free text search can be converted into keyword based search making the retrieval more efficient.
- The repository can be extended further by adding more components in the categories present and also by adding new and relevant components.
- The repository and interface can provide for COTS components so that the user can use the components as such, saving lots of time and effort.

References

[1] Falls Church, "Software Reuse Executive Primer", Department of Defence, April, 1996.

[2] Don Navso, "Software Reuse Guide", Department of the Navy 1995, pp-2.

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

[4] H.Mili / A.Mili, "Reuse Based Software Engineering: Techniques, Organization and Measurement", wiley- Interscience publication, 2000, pp- 8-9.

[5] H.Mili / A.Mili, "Reuse Based Software Engineering: Techniques, Organization and Measurement", wiley- Interscience publication, 2000, pp- 531.

[6] H.Mili / A.Mili, "Reuse Based Software Engineering: Techniques, Organization and Measurement", wiley- Interscience publication, 2000, pp- 447- 450.

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

[8] Scott Henninger, "ACM Transactions on Software Engineering and Methodology, Vol. 6", April 1997, pp-116-120.

[9] G.Caldiera and V.R.Basili, "Identifying and Qualifying Reusable Software Components Computer", 2000, pp- 61–70.

[10] S.Henninger, "Using Iterative Refinement to Find Reusable Software", IEEE software,1994, pp- 11,5.

[11] C.Fox, W.B.Frakes and R.Baeza-Yates, " Lexical Analysis and Stoplists. In Information Retrieval: Data Structures and Algorithms", Eds. Prentice-Hall, Englewood Cliffs, N.J.,1992, pp- 102–130.

[12] B.Raghunathan/ B.Sankaran, "Management of Enterprise Multimedia Information",vol. 4, 2006.

[13] Y.Ye and G.Fischer, "Supporting Reuse by Delivering Task Relevant and Personalized Information" In *ICSE* 2002 - 24th International Conference on Software Engineering, Orlando, Florida, USA, 2002, pp- 513-523

[14] P.W.Foltz and W.Kintsch, “An Empirical Study of Retrieval by Reformulation on HELGON. Tech.”, Inst. of Cognitive Science, Univ. of Colorado, Boulder, Colo., 1988, pp- 88-9

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

[16] W.B.Frakes, and P.B.Gandel, “Representing Reusable Software”, Software. Tech.32, 1997, pp- 653–664.

[17] B.Curtis, T.J.Biggerstaff and A.J.Perlis, “Cognitive Issues in Reusing Software Artifacts”, Software Reusability.Vol. 2, 1989, pp- 269–287.

[18] R.Prieto-Dı´Az, and G.Arango, “Domain Analysis and Software Systems Modeling”, IEEE Computer Society Press, Los Alamitos, Calif., 1991.

[19] G.Salton, and M.J.Mcgill, “Introduction to Modern Information Retrieval”, McGraw Hill, New York, 1983.

[20] W.B.Frakes, and T.Pole, “An Empirical Study of Representation Methods for Reusable Software Components”, IEEE Trans. Software. Eng.,1994, pp-617–630.

[21] D.C.Blair, And M.E.Maron, “An Evaluation of Retrieval Effectiveness for a Full-text Document-Retrieval System”, Commun. *ACM*, 1985, pp- 289–299.

[22] Navneet Kaur, **“**Retrieving Best Component From Reusable Repository”,2005, pp- 23.

[23] J M Spivey, “An Introduction to Z and Formal Specifications”, January 1989.

[24] Jun-Jang Jeng, Betty H. C. Cheng, “Specification Matching for Software Reuse: a Foundation”,ACM, 1995, pp 97-105.

[25] Zhu Jingbo, Yao Tianshun “A Knowledge-Based Approach to Text Classification”, 2001.

[26] Hawisher, Gail E., Paul LeBlanc, Charles Moran, and Cynthia L. Selfe, “Computers and the Teaching of Writing in American Higher Education”, A History Ablex Publishing Corporation, Norwood NJ, 1996, pp- 213.

[27] Panos Constantopoulos and Martin Dorr, “Component Classification in the Software Information Base”, Prentice Hall, 1995, pp- 177-200.

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

[29] Oualid Khayati, Jean-Pierre Giraudin, “Components Retrieval Systems”, 2001, pp-113.

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

[31] Cem Kaner, “Quality Assurance in software testing”, Florida Institute of Technology, Orlando, FL, November 2006.

[32] Binder, V.Robert, “Testing Object-Oriented Systems: Objects, Patterns, and Tools”, Addison-Wesley Professional, 1999, pp- 45.

[33] Beizer, Boris, “Software Testing Techniques”, (Second ed.). New York: Van Nostrand Reinhold, 1990, pp- 21,430.

[34] “IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries”. 1990, pp – 343.

[35] Van Veenendaal, Erik, “Standard Glossary of Terms used in Software Testing”, June 2010.

[36] Maurizio Lenzerini, “Data Integration: A Theoretical Perspective”, PODS, 2002, pp- 233–246.

Research Publications

Research Paper Communicated

Divya Pandove, Shivani Goel, “Design and Implementation of an Interface for Effective Retrieval”, communicated in Knowledge and Information Systems Journal by Springer. (June, 2010).