Implementing Improved Classification and Multiple Search Criteria in Software Repository

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

Master of Engineering
in
Software Engineering

Submitted By
Vaneet Kaur Bhatia
(800931022)

Under the supervision of:
Ms. Shivani Goel
(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "***Implementing Improved Classification and Multiple Search Criteria in Software Repository***", 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 *Ms. Shivani Goel* and refers other researcher's work which are duly listed in the reference section.

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

Vaneet
(Vaneet Kaur Bhatia)

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

Shivani Goel
(Ms. Shivani Goel)
Assistant Professor,
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Head 30/6/11
Computer Science and Engineering Department
Thapar University
Patiala

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

Acknowledgement

I would like to acknowledge everyone who supported me during my experience at Thapar University and my work on this thesis.

Firstly, I would like to thank my guide, Ms. Shivani Goel, Assistant Professor, CSED, Thapar University, Patiala for her constant support, time, patience and guidance. Her feedback and editorial comments were also valuable for writing this thesis. It was a great opportunity to work under her supervision.

Then I would like to thank Dr. Maninder Singh, Head of the Department, CSED, Thapar University, Patiala for providing all the facilities and environment. I would also like to thank all my Teachers for their stimulating discussion and invaluable suggestions during the period of my work.

I would also like to thank my parents who stayed with me and supported me all the time while I made this dream come true.

Finally I would like to thank my friends especially Deepak, Arpita Sharma, Amandeep Kaur Johar, Ravneet Kaur Grewal, Ravneet Kaur Chawla, Aradhana Majithia, Ipneet Kaur, Aarti Sharma and Sonam Chawla for their support.

Last but not the least I would like to thank God who have always been with me in my good and bad times.

Vaneet Kaur Bhatia

(800931022)

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 purpose as well. The problem that is generally encountered by much of the research and development in information retrieval is aimed at improving the effectiveness and efficiency of retrieval.

Repositories should be designed to meet the growing and changing needs of the software development organizations as it is used for retrieving large numbers of software components. Storage and representation of reusable software components in software repositories to assist retrieval is a key concern area. Effective component retrieval from a large repository is one such aim of the ongoing research. In software component repository thousand of components are stored using various classification techniques.

In this thesis various assets of the component repository like component searching mechanisms and classifications such as Free Text, Enumerated, Attribute Value, and Faceted classifications are discussed. A New classification scheme is proposed which covers the limitations of existing classification schemes.

In order to do so, the first requirement is a user friendly interface which provides the best solution for reorganization of the components in the repository so that recall and precision may be maximized. A component repository system is built in which the users can perform search based on multiple search criteria. The basic comparison of search presented in this thesis on the basis of precision and recall deals with evaluating the 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 Component Based Software Engineering

The component-based software engineering (CBSE) or component-based development (CBD) emphasizes the development of applications based on components so that the applications are easy to maintain, and extend. A component is an independent and self-sufficient part of a system having complete functionalities.

CBSE is a process that aims to design and construct software systems using reusable software components. Component-Based Software Engineering (CBSE) refers to the construction of large software systems from existing parts. This approach to code reuse reduces the production costs and enhances the maintainability of the software system [9].

CBSE is different as compared to other development strategies in a sense that when a component is developed it is intended that it can be integrated with other systems. The main idea of using component-based development is reusability. The biggest challenge facing component based software reuse is managing a large number of stored reusable components efficiently to allow fast allocating and retrieving [10].

1.1.1 Need of CBSE

The goal of component-based software engineering is to increase the productivity, quality, and time-to market in software development. CBSE uses software engineering principles to apply the same idea as object oriented programming to the whole process of designing and constructing software systems. It focuses on reusing and adapting existing components, as opposed to just coding in a particular style. CBSE encourages the composition of software systems, as opposed to programming them [25]. The software systems built using CBSE are not only simple and cheaper but usually turn out to be more robust, adaptable and updateable. CBSE allows use of predictable architectural patterns and standard software architecture leading to a higher quality end result. Component-based systems emphasise appropriate reuse and composition of software components as a key concept. However, 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 [31,32].

1.1.2 A Software component

Brown and Wallnau describe a component as "a nontrivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well defined architecture" [2].

Brown and Wallnau describe a Software component as "a unit of composition with contractually specified and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" [2].

1.2 Need of reusability in CBSE

The Component-Based Development, also called development "with reuse", deals with the applications construction. These applications are composed by reusing existing components. If needed, new components can be developed, or even acquired from third party COTS *(Commercial Off-The-Shelf)* .COTS comes in a variety of types and levels of software, e.g., components that provide specific functionality such as subroutines, classes, frameworks, and even complete applications or tools used to generate code such as domain-oriented language processors and application generators. Many companies are providing reusable off-the-shelf components for specific domains, and those components can be purchased by developers from the market. As this trend continues, programmers may be able to create their own systems in the future by integrating components from different component vendors.

Software Reuse is the process of implementing or updating software systems using existing software assets [12].Anything that is produced from a software development effort can potentially be reused. It is commonly agreed that software reuse does not only concern code but reuse of a software object implies the concurrent reuse of objects associated with it, plus information about the objects. Thus, designs requirements specifications, development processes, and decision experiences are also candidates for reuse [13]. In this we use taxonomy to describe and compare the different approaches and make generalizations about the field of software reuse. The taxonomy characterizes each reuse approach in terms of its reusable artifacts and the way these artifacts are abstracted, selected, specialized, and integrated [34]. Reusable software components have both short-

term and long-term benefits for the development of software systems. Short-term benefits are the immediate benefits that a programmer achieves during the implementation of a programming task. In long-term benefits the programmer does not reuse the components, but they extend to the whole life cycle of the software system and to later programming activities of the programmer.

Software reuse is the use of existing software or software knowledge to construct new software [14]. Reusable assets can be either reusable software or software knowledge. Reusability is a property of a software asset that indicates its probability of reuse. The main purpose of software reuse is to improve software quality and productivity.

1.3 Repository

Repository commonly refers to a location for storage, often for safety or preservation. In the simplest term, a repository is nothing more than a collection of knowledgeable assets. A repository is a place where the data is stored. It is often stored in relational databases, but can be stored anywhere that can hold the data: Excel spreadsheets, hierarchical databases, or even text files.

A relational database is a distinct type of database where data is stored in related tables. Not all repositories are stored in relational databases, but many are, for example an XML repository is a system for storing and retrieving XML data.

A repository is a place where data is stored and maintained. It can be, e.g., a place where digital data is stored, a site where eprints are located, a place where multiple databases or files are located for distribution over a network [15].

1.3.1 Types of repositories

Repository commonly refers to a location for storage, often for safety or preservation. It may also refer to:

- Repository clone, concept from distributed revision control.
- Repository Open Service Interface Definition, specification which defines the storing and retrieving of digital content.
- Component repository management, field of configuration management.
- Deep geological repository for radioactive waste.

- Information repository, developed to mitigate problems arising from data proliferation.
- Institutional repository, universities create digital collections of an institution's scholarly research.
- National repository, repository for academic publications.
- National Mine Map Repository, U.S. government-run facility assuming the responsibility of storing information extracted from mining maps, including pictures of the actual maps.
- Serum repository, facility which stores frozen serum for future retrieval and study.
- Software repository, storage location from which software packages may be retrieved and installed on a computer.
- Digital repository, also known as a virtual or digital library.

1.4 Introduction to software repositories

A *software repository* is a storage location from which software packages may be retrieved and installed on a computer. Many software publishers and other organizations maintain servers on the Internet for this purpose, either free of charge or for a subscription fee. Repositories may be solely for particular programs, such as CPAN for the PERL programming language, or for an entire operating system. Operators of such repositories typically provide a package management system, tools intended to search for, install and otherwise manipulate software packages from the repositories [15]. It archives many software systems with their source codes:

- In open source community
    - Provide platforms for many open source projects
    - E.g. SourceForge (http://sourceforge.net/)
- Industrial context
    - Archive software systems created *in a company*
    - To share information about projects that exist (or existed) in the company
    - Useful especially for large and distributed organization
    - E.g. Corporate Source, Progressive Open Source

Software repositories contain a wealth of valuable information about the evolution of a software project.

1.5 Software repository and its importance in software reuse

Repositories play a pivotal role in an integrated reuse- based application development environment. Reusing software components implies their persistent storage and maintenance, and the ability to efficiently find them. Repositories built with reuse in mind can be considered as special-purpose information systems, required to support powerful semantic modeling, flexible retrieval of varied software descriptions of multimedia nature, and efficiency optimization directed towards a large variety of classes rather than large populations per class. Software repository can be seen as an *information system* where users can access the software components and their related information using a retrieval subsystem. Software artifacts are stored in a repository which supports their efficient selection. Comprehending their functionality and usage before adaptation and subsequent use in the composition of a new application is also recognized as essential part in the reuse process. Reuse based development reuse targeted repository illustrated in Figure 1.1 plays a critical role associated with its function and tools. Other distinguishing characteristics are its capacity for flexible retrieval and browsing of multimedia data and the optimization of efficiency for network structures consisting of a large variety of classes rather than relatively few classes with large populations per class.

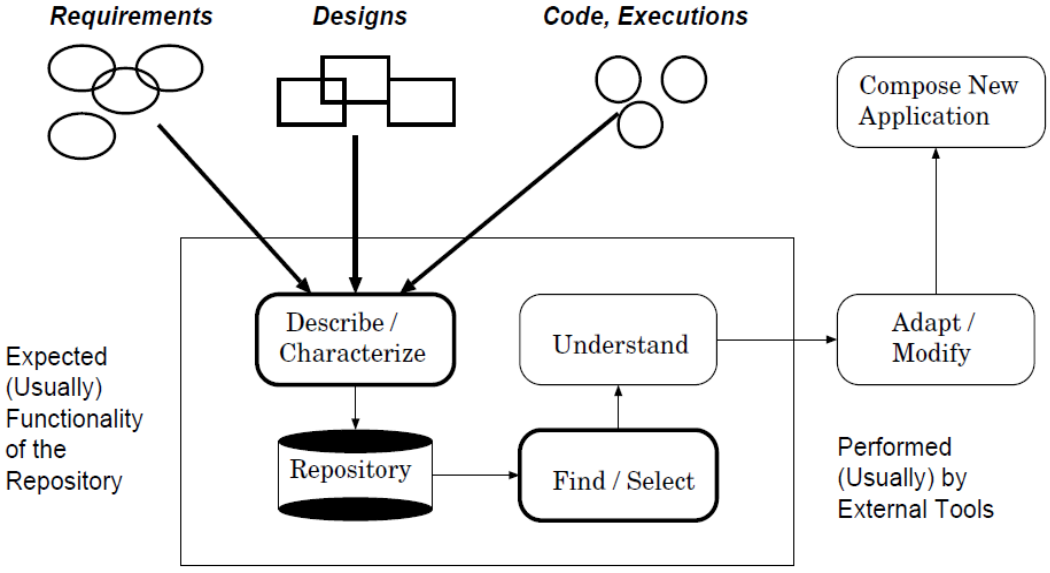

Figure 1.1: Reuse based development cycle [14]

This methodology is used for organizing the reusable components in the repository. It is commonly agreed that software reuse does not only concern code. It also implies the concurrent reuse of objects associated with it, plus information about the objects. Thus, designs requirements specifications, development processes, and decision experiences are also candidates for reuse [14].

1.5.1 Repository: Component Reuse

An independent information repository system (or component repository system) can be abstractly thought of as secondary information storage from the perspective of computer users because information stored in this is accessible only after users have stopped working on their current tasks and switched from their workspaces.

The term repository may be used for different purposes. Among them, we highlight:

- The storage of models

A development process involves the generation of many artifacts that are normally stored in different places. A repository allows the storage centralization of these artifacts and the control of each one of their versions [27].

- Integration of tools

The models generated during a development process often originate from distinct tools. For this a language is needed to communicate. The repository may serve as a temporary storage and the intermediation between the tools, that is, it may transform the entrance model of one of them into a model compatible to the other.

- Maintenance of Legacy Systems

A repository can be useful also for the storage of data related to old systems. This data can be analyzed later to determine the possible impacts generated by modifications.

1.6 Organization of the report

The chapters in thesis are organised as follows:

Chapter 2 This chapter describes in detail the literature survey about software repository and its various classification schemes.

Chapter 3 This chapter describes gap analysis and various objectives to be achieved.

Chapter 4 This chapter describes in detail the solution of problem, proposed model and the design of component repository system. It defines the basic features along with the construction of the repository. It further goes on to explain the working of the system.

Chapter 5 This chapter deals with the experimental results obtained of the system on the basis of precision and recall.

Chapter 6 This chapter discusses conclusion and future scope of the work done. And at last are the references and paper communicated.

Chapter 2

Literature Review

Previous chapter gave a brief introduction about CBSE and repository. This chapter gives a brief introduction about software repository, its facets and various classification schemes.

2.1 Software Component Repository

A software component repository represents a potential tool for the exercise of reuse inside organizations. When the components stored in repository are well organized and managed, this potential becomes a reality. Through a repository sustaining the reuse politics inside the enterprise, aiming at the benefits of this practice, there will be an increase in the levels of productivity and quality in the process of development of the organization using it. This means that software reuse is a business strategy that should be sustained, in a technological point of view, by a corporative components repository [27].

2.1.1 Need of Component Repository

Practice of reuse could bring many benefits to an organization. For this it is necessary that all members of the organization know the components developed so they can evaluate them and possibly, choose for their reuse in other projects. The main intention is to enable the projects to be composed of small components in the future. Therefore, this would decrease the time for delivering the product to the client and increasing the productivity and quality of these projects.

The main idea is to allow the repository to serve as a knowledge base, since a software component is the product of tasks based on the intensive use of knowledge. Beyond software artifacts, as the components font code, a repository may also be called to manage other types of documents, such as proposals, requisites specification, memorandum, etc that is, all the information used as input and as sub product of the different software development activities [27].

2.1.2 Construction of a Repository

The idea of constructing new software by composition from collection of reusable components is not new and clearly has many attractions. However it has yet to receive widespread implementations. Several reasons for this are:

- The repository must be large enough to contain a useful collection of components, yet each component must be readily available.
- The components must be highly reliable since they will be reused in many applications.
- There are some means for extracting components from existing code for addition to the archive: writing a complete library of components from scratch would involve a great deal of investment of effort before any return on the investment would be perceived.
- In order to be widely useful the components should be written to handle the most general cases, this means that programs constructed from components can be much less efficient that programs constructed from components can be less efficient than programs written from scratch which can exploit regularities in the data.

2.2 Facets of Software Component Repository

The software repository is used for storing, managing, and retrieving large numbers of software components. Repositories should be designed to meet the growing and changing needs of the software development organizations. Storage and representation of reusable software components in software repositories to assist retrieval is a key concern area. In this thesis various assets of the component repository like component searching mechanisms and classifications such as Free Text, Enumerated, Attribute Value, and Faceted classifications has been discussed. Developers and end users can formulate high-level, aspect-based queries to retrieve components according to their needs.

2.2.1 Component Repository

An independent information repository system (or component repository system) can be abstractly thought of as secondary information storage from the perspective of computer users because information stored in this is accessible only after users have stopped working on their current tasks and switched from their workspaces. The retrieval process

finds the components that match given reuse queries. An effective retrieval mechanism including a representation schema for indexing and a matching criterion between a query and a component is essential. The structure of a repository is generally regarded as 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 [7].

2.2.2 Storage and Retrieval of Software Component

Reusability is a process of utilizing and applying already developed components. So there are many work products that can be reused, for example source code, designs, specifications, architectures and documentation. Successful reuse requires having a wide variety of high quality components, proper classification and retrieval mechanisms. Effective software reuse requires that the users of the system have access to appropriate components. Retrieval should allow users to formulate high-level queries about component capabilities and takes account of the context in which a query is performed to assist query formulation [1]. The user must access these components accurately and quickly, and if necessary, be able to modify them. Classifying software allows reusers to organize collections of components into structures so that they can be searched easily. Most retrieval methods require some kind of classification of the components. Four different classification techniques had been previously employed to construct reuse repository namely Free Text, Enumerated, Attribute Value, and Faceted classifications.

2.2.3 Querying and Browsing component

Querying and browsing are the two major information access mechanisms for most users. Querying is direct that users formulate a query and the system returns information matching the query. However to formulate a query is a quiet difficult task because users have to overcome the gap from the situational model to the system model. In browsing, users determine the usefulness or relevance of the information currently being displayed in terms of their task and traverse its associated links Mili et al. [8] claims that browsing is the most predominant pattern of component repository usage because most programmers often cannot formulate clearly-defined requirements for reusable components so they rely on browsing to get familiar with available reusable components in the repository.

2.2.4 Effective retrieval mechanism in retrieving

Several retrieval techniques are produced till now. There are three major approaches in retrieval mechanisms text-based, descriptor-based and formal specification-based. In text-based approach a component is represented by their textual documents and information retrieval technology is used to match components to queries. In descriptor-based approach, components are represented by a set of selected descriptors [16]. The semantic relationships among those descriptors are captured in a predetermined structure that can be specified by a semantic network. In specification-based approach, components are represented with formal specification languages, and specification refinement systems are used to determine whether a component matches a query written in formal specification languages or not. Retrieval mechanisms play an important role in locating reusable components that match reuse queries.

2.2.5 Evaluation

To check whether component repository works effectively we evaluate the correctness and effectiveness by using two measures that are recall and precision. Precision and recall are two concepts that have traditionally been used to evaluate the method of retrieval software components.

$$\text{PRECISION} = \frac{\text{Ratio of relevant retrieved assets}}{\text{Total number of retrieved assets}}$$

This is a number that ranges between 0 and 1.Under the hypothesis that all the library assets are visited, 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 no assets are returned.

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

$$\text{RECALL} = \frac{\text{Ratio of relevant retrieved assets}}{\text{Total number of relevant assets in the library}}$$

This is the number that ranges between 0 and1.Under the hypothesis that all the library assets are visited, we get perfect recall (=1) whenever the relevance criterion logically implies the matching condition. This can be achieved in particular by letting the matching

condition be true which means that all library assets are returned. Recall means to get all the relevant components.

It is very difficult to get both high precision and high recall using the classical approach. For effective use, the keywords have to be independent, they have to span the range of the users need, and the users must be able to pick the right keywords.

2.3 Approaches for software component classification

Classifying software allows reusers to organize collections of components into structures so that they can be searched easily. Most retrieval methods require some kind of classification of the components. Four different classification techniques had been previously employed to construct reuse repository namely Free Text, Enumerated, Attribute Value, and Faceted classifications.

2.3.1 Free Text classification

The retrieval system of free text classification is typically based upon a keyword search. In this type of searching technique, a user inputs keywords to search and as a result a ranked list of documents is returned. We can see a number of search engines using keyword search technique to search documents on Internet. As we know search engines like Google, Yahoo, MSN search, and AltaVista are some well-known web search engines that support keyword-based searching as a basic searching technique to search documents and websites etc. Free-text classification is also referred to as uncontrolled vocabulary, consists in analyzing word frequencies in natural text [4]. Relevant keywords are derived automatically by their statistical and positional properties, thus resulting in what is called automatic indexing.

Two processes are involved in keyword search that is indexing and searching. The indexing process looks into all the available documents in different repositories or databases and creates a list of search items, which could be used for search purpose. 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 documentation is not necessary for software component repositories than for text documents.

Challenges faced in Free Text classification

- Keyword search gives freedom to users to freely submit any query to search engines; however this freedom may rise following two problems as:
    - Recognize keywords that best explain the need of users.
    - Possible ways in which a user may search.
- Another problem is that it would be inaccurate to characterize most source code as being documented adequately for these methods.
- The vocabulary becomes limited, freezing chances of innovation by the user.
- Retrieval effectiveness of free-text methods has been questioned within text-intensive domains [28].

2.3.2 Enumerated Classification

Enumerated classification is a single dimension classification which uses a set of mutually exclusive classes. In enumerated classification a subject area is broken into mutually exclusive, usually hierarchical, classes [30]. An example of this is the Dewey Decimal system used to classify books in a library [3]. Each subject area, e.g. Physics, Chemistry etc, has its own classifying code. As a sub code of this is a specialist subject area within the main subject. These codes can again be sub coded by author. 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. This classification scheme is the ability to iteratively divide an information space into smaller pieces that reduces the amount of information that needs to be pursued [29]. A partial expansion of the Dewey hierarchy is shown in Figure 2.1

Structured Systems Programming could match any of the following classes: 001.61 (systems analysis), 001.642 5 (software), 003 (systems), 620.72 (systems analysis), or 620.73 (systems construction). Implicit in enumerative classification is the expertise of the librarian in both the classification scheme and the subject matter or domain of the title. Deciding which the most appropriate class is a difficult task. The title in this example is entered under 001.642 5 (software) [25].

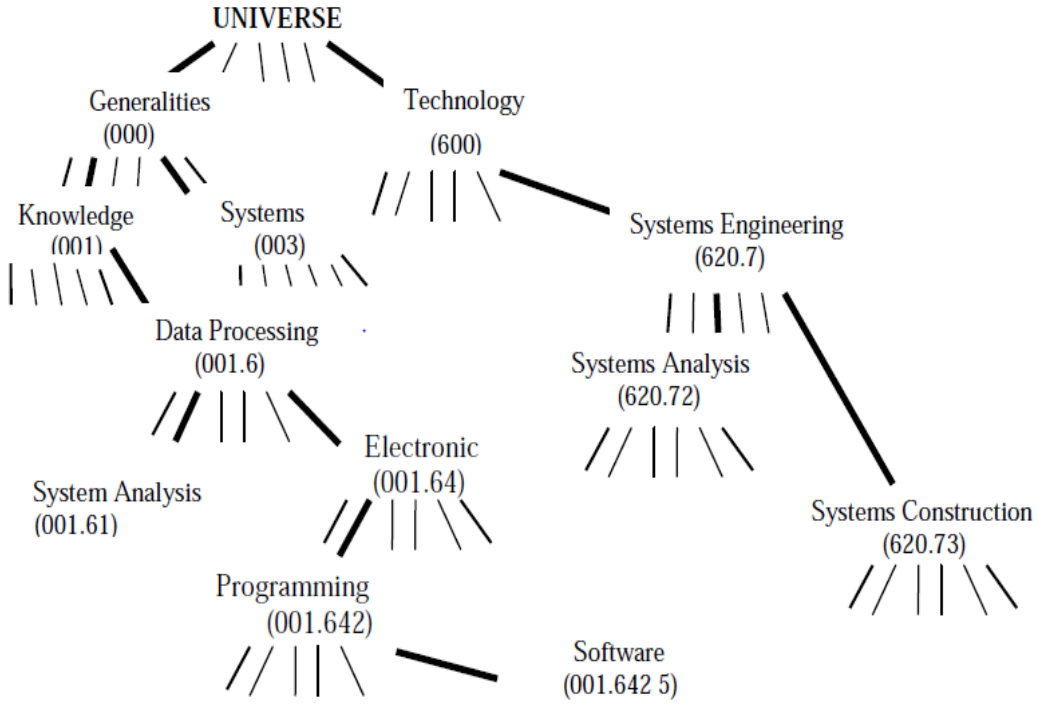

Figure 2.1: Partial Expansion of the Dewey Decimal Hierarchy [25]

Challenges faced in Enumerated classification

- Cross-references establish links among related classes to simplify the class selection and to facilitate searching, but they add a complex network of relationships that is difficult to represent and maintain [17].
- Major problem is that this type of classification schemes as is one dimensional will not allow flexible classification of components into more than one place for reusable software components.
- It does however; provide substantial support for best effort retrieval of components.It requires users to understand the structure and contents of repository to effectively retrieve the information.
- There is inherent inflexibility present in this scheme. There is trouble in distinguishing between class labels such as Proceedings and In proceedings in a database of computer science literature references [17].

2.3.3 Attribute Value Classification

This type of classification scheme uses a set of attributes to classify a component .For example; a book has many attributes such as the author, the publisher, title and a unique

ISBN number and classification code in the Dewey Decimal system [5]. Then we see the requirement and on that basis the attributes could be concerned with the number of pages, the size of the paper used and the publishing date etc. The attributes related to a book can be:

- Multidimensional. The book can be classified in different places using different attributes.
- Bulky. All possible variations of attributes which may not be known at the time of classification [3].

Challenges faced in Attribute Value classification

- Each attribute has the same weighting as the rest, the implications being that it is very difficult to determine how close a retrieved component is to the intended requirements, without visually inspecting the contents [3].
- It is the slowest method and has no ordering.

2.3.4 Faceted Classification

Faceted classification so known as faceted navigation or faceted browsing was proposed by Prieto-Diaz and Freeman in 1987[4] that relies on facets which are extracted by experts to describe features about components. Features serve as component descriptors, such as the component's functionality, how to run the component, and implementation details .Like the attribute classification method, various facets classify components however there are usually a lot fewer facets than there are potential attributes. Sometimes faceted search is also referred to explorative search and guided search, because users are given choice to select features available for search. This helps the users to accomplish his search goals rapidly and efficiently.

Faceted classification and retrieval has proven to be very effective in retrieving reuse component from repositories, but the approach is labor intensive. Ruben Prieto-Diaz has proposed a faceted scheme that uses six facets.

- The functional facets are: Function, Objects and Medium.
- The environmental facets are: System type, Functional area, setting [5].

Each of the facets has to have values assigned at the time the component is classified.

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 [18].
- Faceted schemes are more effective for domain-specific collections than for broad, heterogeneous collections. A faceted scheme for a diversified collection becomes too general, losing its descriptive precision [4].
- 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 [19].

2.4 Browsing and Searching

Searching and browsing have long served as the principal techniques for software developers to locate components from component repositories.

Searching is direct and fast. Software developers formulate a query, and the repository system returns components that match the query. Formulating queries is a cognitively challenging task because software developers have to overcome the gap from the situational model to the system model (i.e., the description of the components in the repository) [20].

Software developers in browsing determine the usefulness or relevance of the components currently being displayed in terms of their development task, and traverse the associated links in the component repository. Generally people who want some information prefer browsing to searching because they do not need to commit resources at first and can incrementally develop their requirements after evaluating the information along the way. Mili et al. [21] claims that browsing is the predominant pattern of component repository usage because many software developers often cannot formulate clearly defined queries. Instead, they rely on browsing to get acquainted with available components in the repository. Most component repositories are structured according to the inheritance relationship. This structure is suitable for locating components whose super nodes (classes or super-classes) are known, but it is not suitable for locating components based on functionality. Some components with similar functionality are scattered in different deep nodes of the inheritance tree [12]. It is very difficult for

software developers to find and compare all of them in order to choose the most appropriate one. Third, in a large component repository, following the right link requires that software developers have a very good understanding of the structure of the whole repository. Most software developers, especially the less experienced ones, may easily get lost in a complex network of nodes while tracing dozens of links. Experience shows that browsing through large structures can be very frustrating and time consuming. Prieto-Diaz has proposed using facets [2], which are groups of related terms in a subject area.

2.4.1 Approaches based on browsing, searching and querying

Component repository systems that support searching have adopted different retrieval mechanisms. Based on the representation schema of components, they can be divided into text-based, structured representation-based and formal method-based approaches.

Component repository systems that support searching have adopted different retrieval mechanisms. Based on the representation schema of components, they can be divided into text-based, structured representation-based and formal method-based approaches. Formal method-based approaches use either signatures or formal specifications [23] to represent components and queries. Most component repository systems that support browsing, such as the Smalltalk programming environment, organize components according to their inheritance structure. Such a browsing structure requires that software developers have extensive knowledge about the structure of the repository to find a component, and it lacks meaningful road maps as to what the repository contains and how to discover the components that are needed for the current task [24].

Browsing not only speeds up the search for components, but also enables software developers to learn and use unanticipated components that they might miss with the conventional information access mechanisms of browsing and searching.

2.5 Storage and Retrieval of Software Components

Storage and representation of reusable software components in software repositories to facilitate convenient identification and retrieval has been always a concern for software reuse researchers.

A retrieval technique is concerned with the comparison of the representation of a query and the representations of documents 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 documents and queries, but different ways of representation may imply using different retrieval techniques [25].

An effective retrieval mechanism including a representation schema for indexing and a matching criterion between a query and a component is essential [26].

2.5.1 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.

Information retrieval system has three basic issues which are shown in Figure 2.2 that represents a general view of the retrieval process.

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

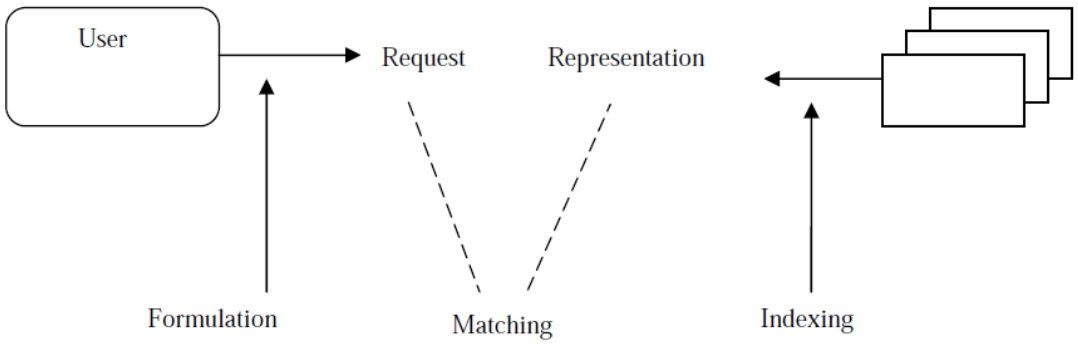

Figure 2.2: A general view of Retrieval Process [25]

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. The searcher knows precisely what he is looking for and where he can find it. For example the title of a book that has to be found in some catalogue. In this case the user is very well able to formulate his information need into a request since he knows very well what he wants.

On the contrary, an intensional information need is much less clear and specific. The user does not know exactly what he is looking for; he usually only has a general idea about his information need. For example, he might be looking for information on a particular subject, but he does not know exactly what kind of information he requires. His information need is thus imprecise and probably incomplete [25].

Indexing

To retrieve any information, this 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.

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 [25]. 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. Here we touch upon the matter of relevance. When a document is ‘similar enough’ to a query, it is ‘relevant’ with respect to that query. Relevance is a very important aim 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.

2.5.2 Challenges faced in the 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.

Chapter 3

Problem Statement

Previous chapters discussed about software repository and its classification schemes and retrieval processes. This chapter focuses on problem statement taken up in the thesis.

3.1 Gap Analysis

Information retrieval from components repository is a tedious work. The size of repository is often very large. Repository contains large number of components and its specification. Repository is a link between development for reuse, where the components are produced, and development with reuse, where components are reused.

For effectively reusing the components from the repository, the selection of proper retrieval technique is essential. Classifications schemes in different repositories have different impact on retrieval. Following table is about various classification schemes and limitations discussed in previous chapter.

Table 3.1: Various classification schemes and their limitations

Classification schemes	Limitations
Enumerated classification scheme	Enumerated classification is the Fast method but is difficult to enlarge.
Free Text classification	Free text classification is uncertain and consists of indexing costs.
Attribute Value classification	Attribute value classification is slowest method and has no ordering.

So a new classification scheme should be considered which will overcome its limitations and enhance the classification efficiency.

3.2 Objectives

Based on the above problem statement, following objectives are required:

- For this there is a need of user friendly interface which provides the best solution for organisation of the components so that it can be reused by the way it is required.
- To propose a new storage retrieval method based on multiple search criteria for better retrieval.
- Developing a repository for the effective retrieval using a new classification scheme.
- To validate the same by calculating precision and recall of repository.

Chapter 4

Design and Implementation of Component Repository System

This chapter discusses about how the problem stated in previous chapter can be solved by designing a new repository named as Component repository system. It is a web application developed in ASP.Net with SQL server at the backend.

This chapter covers proposed model, features, architecture and implementation of the component repository 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 Proposed Model

Existing software components for reuse can be directly classified in the classification scheme into one among the above specified classifications presented in the previous chapter and stored in the reuse repository. Sometimes they need to be adapted according to the requirements. As classification scheme relies on one of the techniques discussed in the previous chapter which shall inherently affect the classification efficiency. New designs of software components for reuse are also subject to classified to classification scheme before storing them in the reuse repository. User will retrieve his desired component with required attributes from reuse repositories. The architecture of proposed system is shown in the Figure 4.1

Proposed classification scheme, which employs a combination of one or more classification techniques, is proposed and likely to enhance the classification efficiency. The proposal is described in the following sub section. This had given rise to development of a software tool to classify a software component and build software component repository.

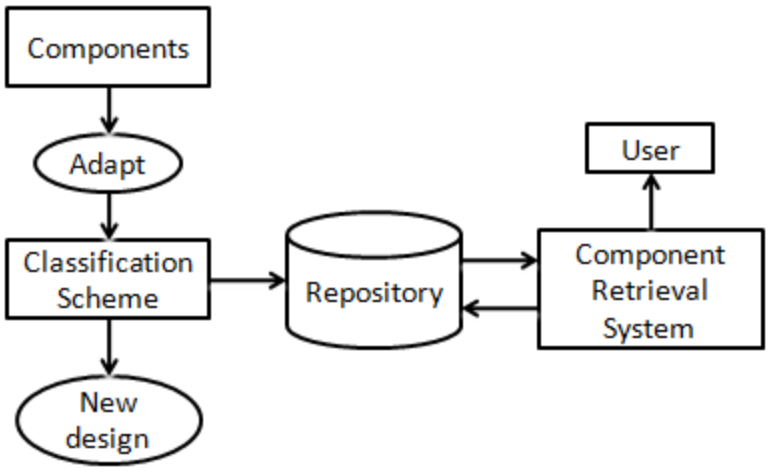

Figure 4.1: Proposed Model

4.1.1 Proposed Classification Scheme

Proposed classification scheme which combines the attribute value and faceted classification schemes to classify components with the following attributes.

- Component Name
- Brand
- Version No.
- Type of Application
- Component Type
- Operating System
- Cost
- Language

The attributes when used in query can narrow down the search space to be used while retrieval. The proposed software tool will provide a user friendly interface for browsing, retrieving and inserting components.

4.2 Proposed Repository

- Easy classification according to multiple search criteria in the repository.
- Selection refinement.
- Both Components and COTS selection

4.3 Features of the Component Repository 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 component keep on changing according to the choice of the user.
- A separate administrative unit is also present; in order to manage various categories, subcategories and user added criteria.

4.4 Construction of the Repository

Construction of the repository here refers to the representation of the components present in it as well as various steps used for its construction.

4.4.1 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. This repository is made on the concept of multiple search criteria. The surrogate in the Component Repository system contains the following fields:

Component Name: Component name is entered here in our system.

Brand: Brand name of component is given here.

Version No: This column gives the various versions of the same component.

Type of application: A dropped down menu contains the type of application a component is having and is selected from this list.

Component Type: Component type of the component is selected here. There are two types of component named as Component and COTS.

Operating system: There are various kinds of operating system present but here the operating system which the component is having is being selected.

Cost: The cost which a component is having is entered here. The cost of the component present here is in US dollars.

Language: A component can be made in any language. Here the language which the component is having is being selected.

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

Last modification date: The date on which the component is last modified is entered here.

4.4.2 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 which can be searched on the basis of multiple criteria. The repository also contains the databases having the surrogates of the 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.

The database manages the administrative unit and the present system has been created with the name dbarepository. The main table named as comp_repository which consists of various fields mentioned earlier in the representation of the components. Figure 4.2 shows the database dbarepository consisting of various fields of table comp_repository as seen in the Microsoft SQL Server Management Studio Express.

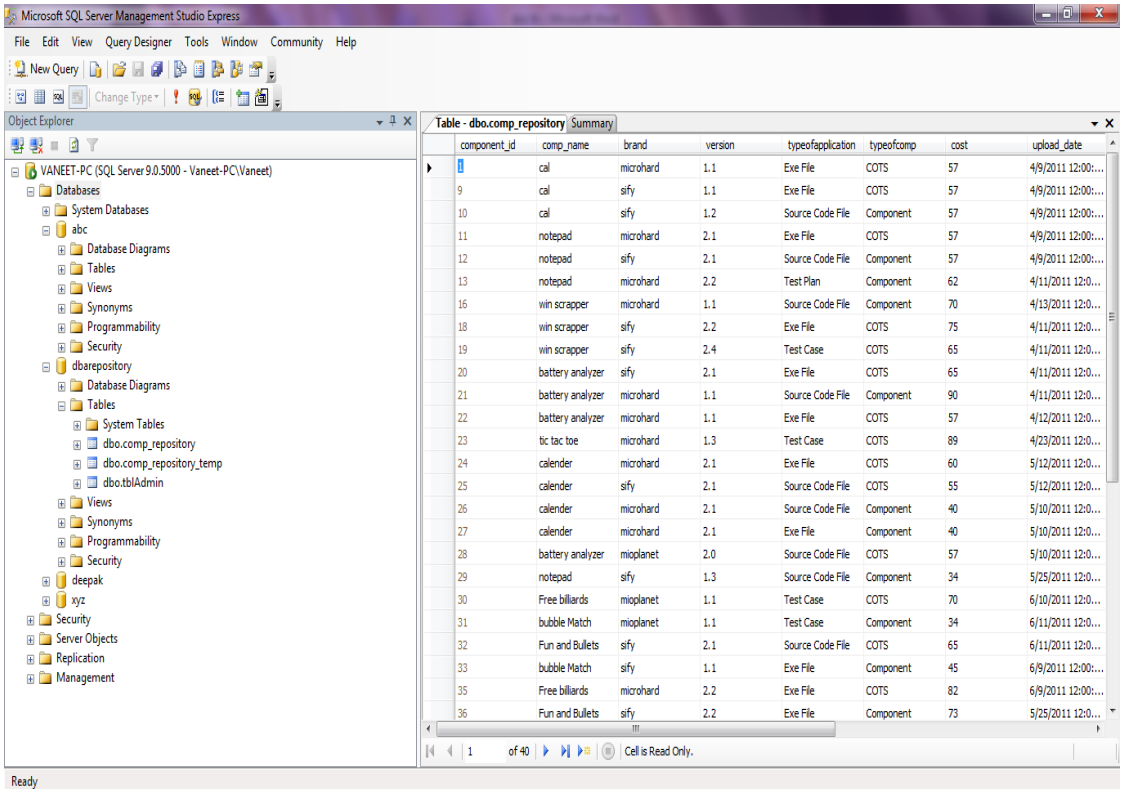

Figure 4.2: Snapshot of the Database 'dbarepository'

4.5 Working of Component Repository System

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

- **User Login**

The user has to login every time they wish to access the site and the components available. Figure 4.3 shows User Login area where the user has to enter his authenticated user name and password to access the system.

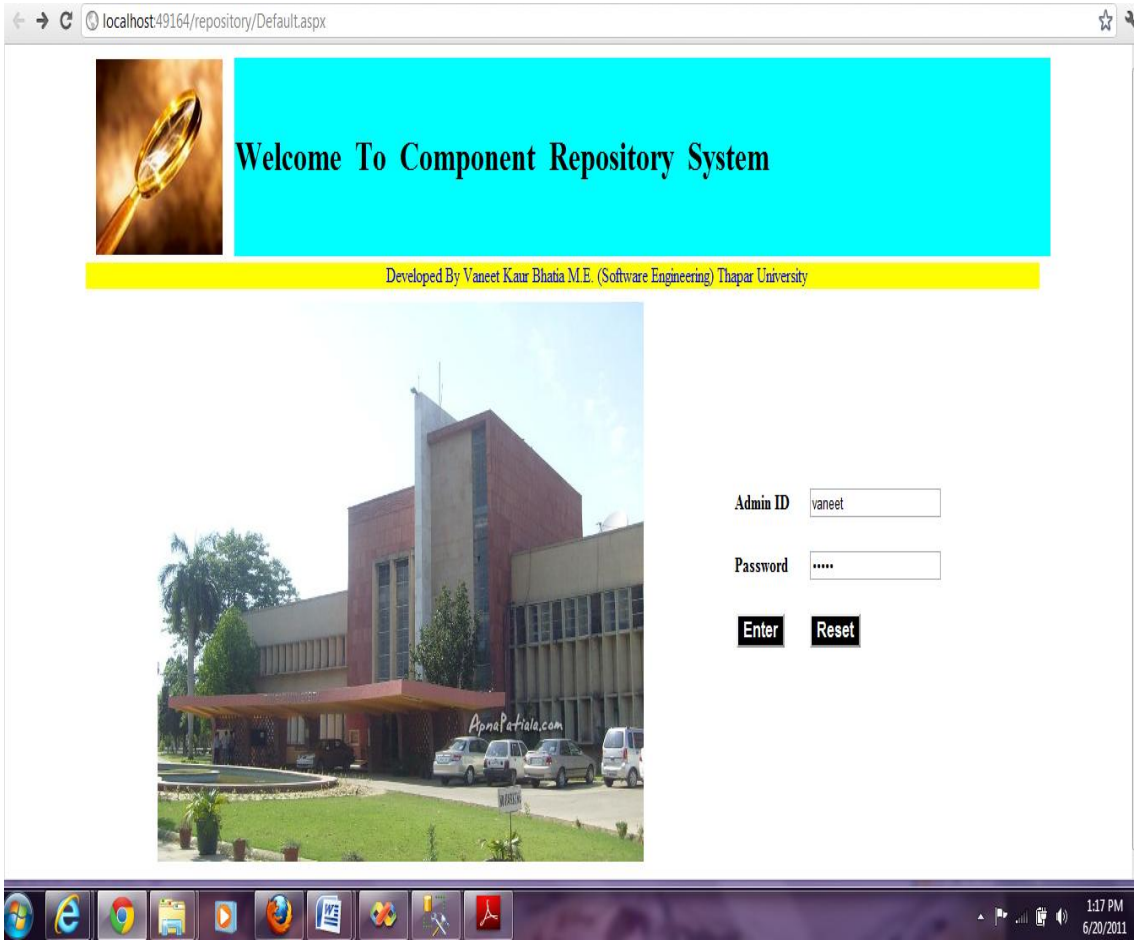

Figure 4.3: Snapshot of User Login Area

- **Main Home page**

This main page is divided into three areas:

- Insert new component: Here the user insert the values of the new component.
- Delete component: Here the user can delete the component according to user.
- Search component: Here the user searches for the component.

Figure 4.4: Snapshot of Main Home page

- **Insert Component**

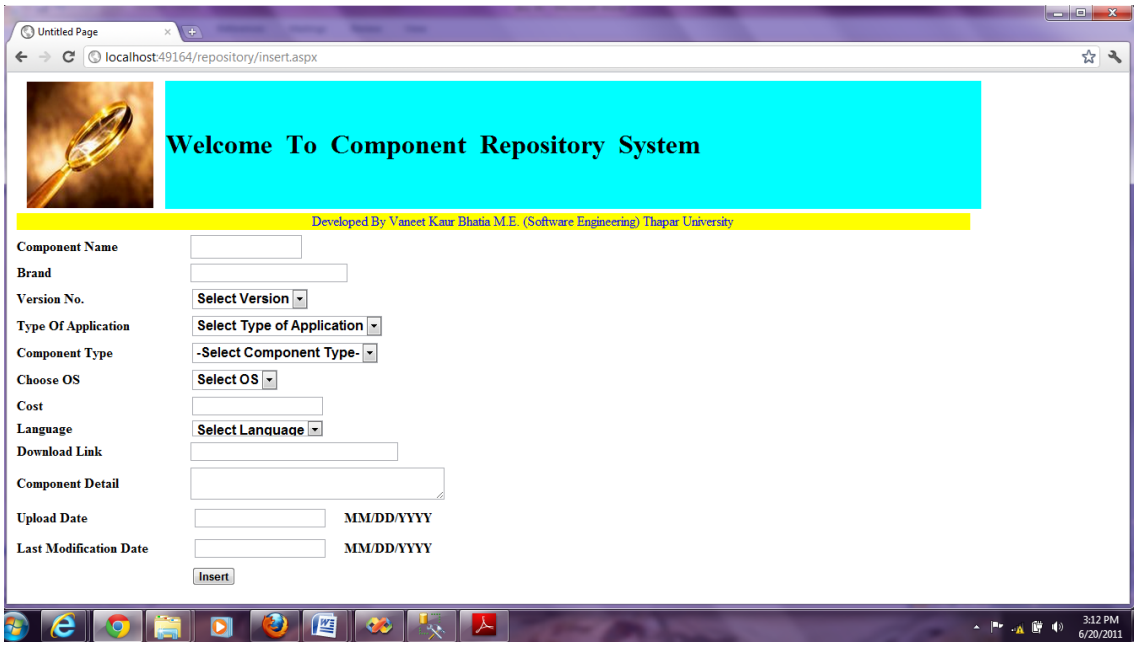

Figure 4.5: Snapshot of Insert Component

- **Delete Component**

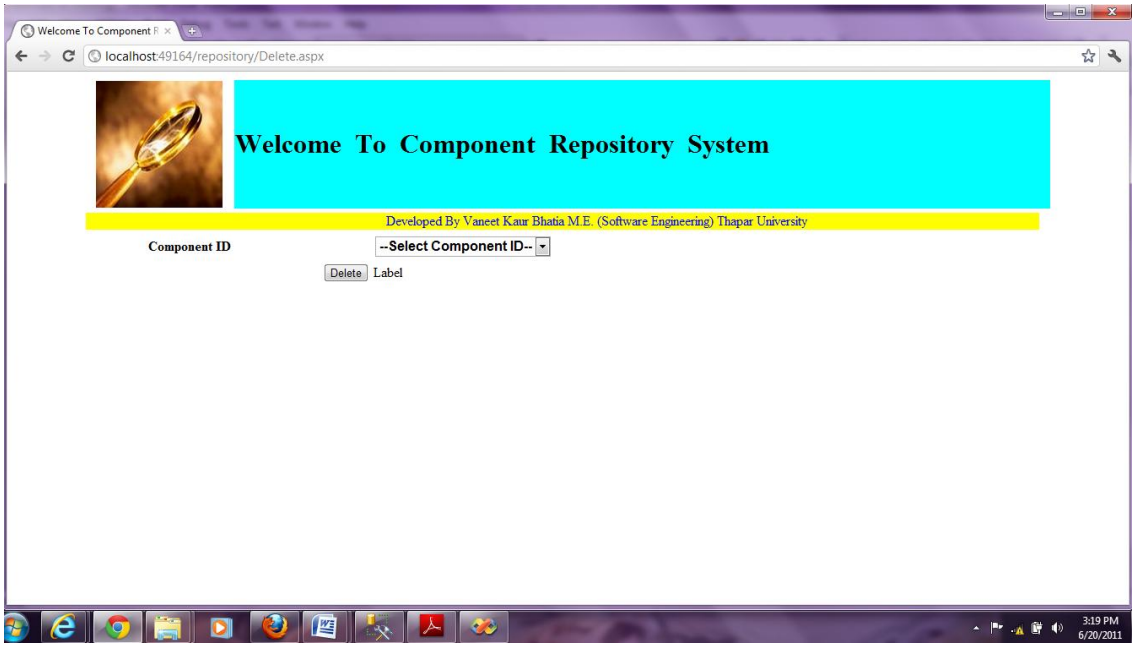

Figure 4.6: Snapshot of Delete Component

- **Search Component**

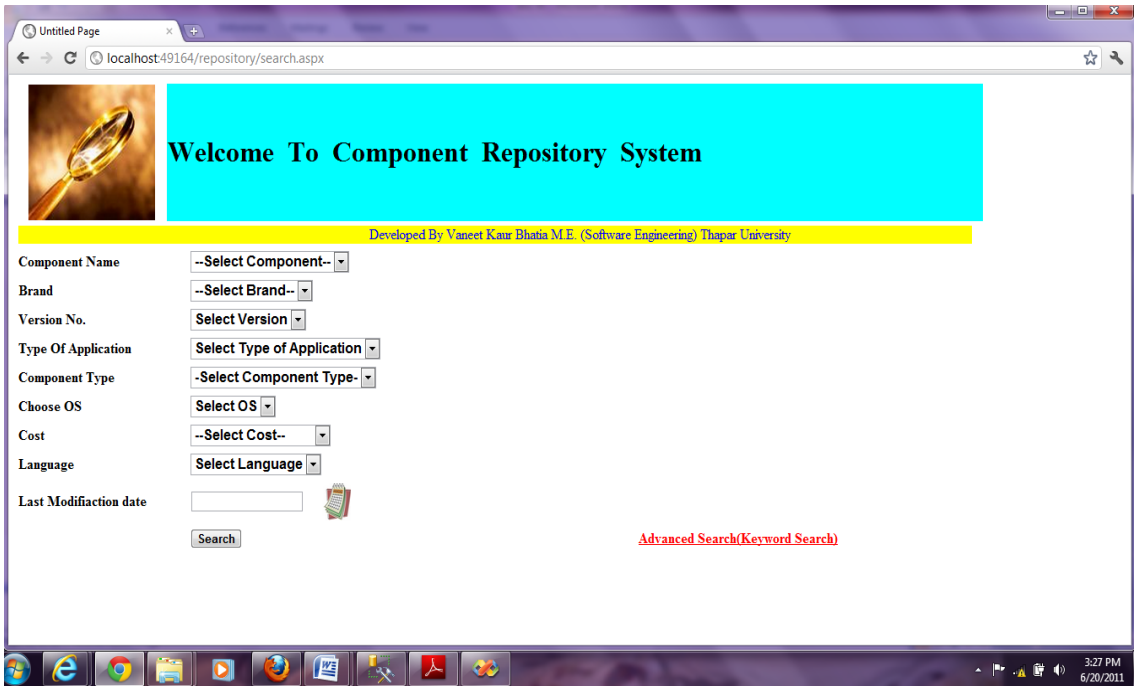

Figure 4.7: Snapshot of Search Component

There are two types of searches which can be performed by the user:

- Query Based search

In this the different values of categories are searched according to the user. The user selects the category and result is displayed on this basis.

- Keyword Based search

In this search the user enters the keyword and on this basis the result is displayed.

- **Query Based search**

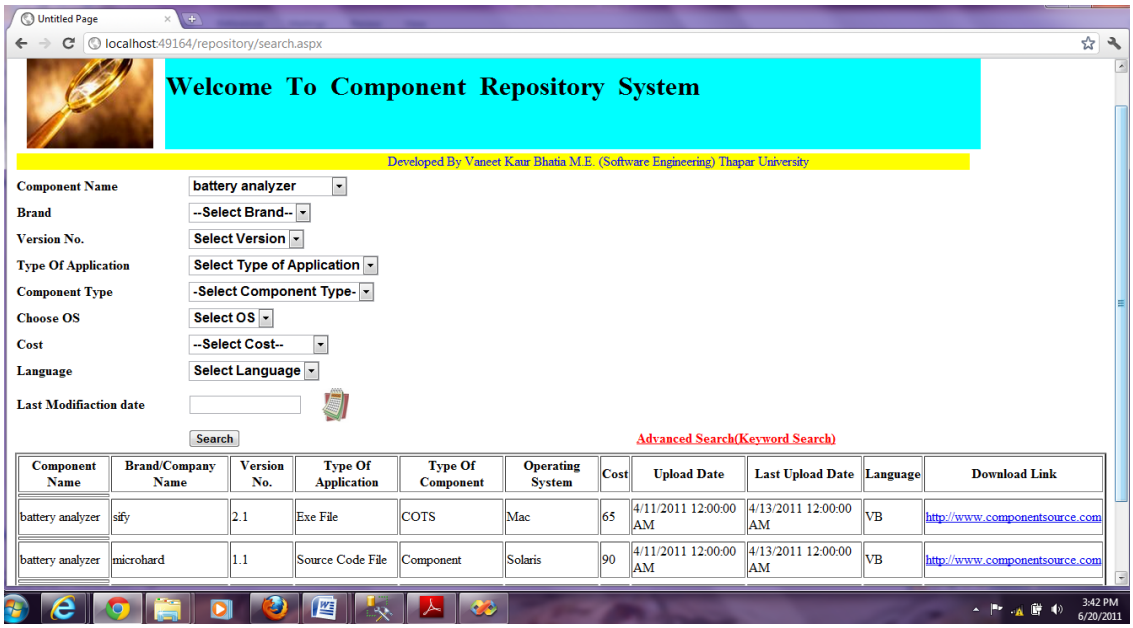

Figure 4.8: Snapshot of Query Based search

- **Keyword Based search**

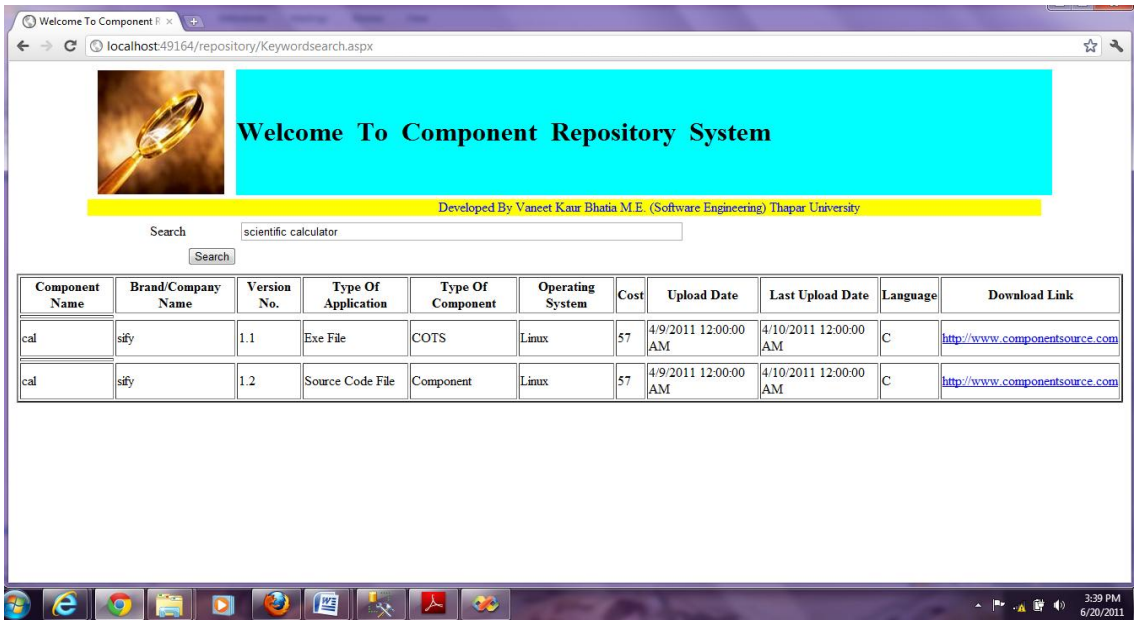

Figure 4.9: Snapshot of Keyword Based search

Chapter 5

Experimental Results

This chapter focuses on the various experimental results based on parameters like precision and recall which has been derived in order to find out the best retrieval technique out of the ones implemented in previous chapter.

5.1 Search criteria of component repository system

In this component repository system the two types of search criteria has been considered.

- Query based search
- Keyword based search

On the basis of these criteria we have calculated the precision and recall of various components.

5.1.1 Query based search

A sample size of 20 components has been taken and on this basis the average precision and recall is calculated.

This search is being categorized in 3 types:

- Search for various applications

Various applications here refer to calculator, notepad, battery analyzer and calendar

- Search for various games

Various games refer to free billiards, tic tac toe, fun and bullets and bubble match

- Search for various browsers

Various browsers refer to Google chrome, Firefox, Internet explorer and safari installer.

For both query based search and keyword search these categories have been considered.

1. Search for various applications such as calculator, notepad, Battery analyzer and Calendar.

Table 5.1: Query based search for various applications

Test										Precision			Recall		
Case	Cname	Brand	Vno	TOA	Ctype	OS	Lang	Cost	RC	RR	TR	P	RR	TR	R
1	Cal	√	x	x	x	x	x	x	Sify	2	2	1	2	3	0.70
2	Cal	√	√	x	x	x	x	x	Sify 1.2	1	1	1	1	3	0.33
3	Cal	x	√	x	x	x	x	x	1.2	1	1	1	1	3	0.30
4	Cal	x	x	√	x	x	x	x	Exe file	2	2	1	2	3	0.70
5	Cal	x	x	x	√	x	x	x	Cots	2	2	1	2	3	0.70
6	Cal	x	x	x	x	√	x	x	Windows	2	2	1	2	3	0.70
7	Cal	x	x	x	x	x	√	x	C	2	2	1	2	3	0.70
8	Cal	x	x	x	x	x	x	√	cost> 50	3	3	1	3	3	1.00
9	Btry Anz	√	x	√	x	x	x	x	Microhard Exe file	3	3	1	1	4	0.25
10	Btry Anz	x	√	x	√	x	x	x	1.1 Cots	2	2	1	2	4	0.50
11	Btry Anz	x	x	x	x	x	√	√	VB cost>50	3	3	1	3	4	0.75
12	Btry Anz	x	x	x	√	x	x	√	Cots cost>50	2	2	1	2	4	0.50
13	Btry Anz	√	√	x	x	x	x	√	Microhard 1.1 cost>50	2	2	1	2	4	0.50
14	Notepad	x	x	√	√	x	x	√	SourceCode Component cost>50	2	2	1	2	3	0.70
15	Notepad	√	x	x	x	x	x	√	Microhard Cost>50	2	2	1	2	3	0.70
16	Notepad	x	√	x	x	√	x	√	2.1 Windows Cost>50	2	2	1	2	3	0.70
17	Notepad	x	√	x	x	√	√	√	2.1 Windows C++ cost>50	2	2	1	2	3	0.70
18	Notepad	√	x	√	x	x	x	x	Sify SourceCode	1	1	1	1	3	0.30
19	Calendr	√	x	x	x	x	x	x	Microhard	3	3	1	3	4	0.75
20	Calendr	√	√	x	x	x	x	x	Microhard 2.1	3	3	1	3	4	0.80
Average										1			0.61		

Graphs of precision and Recall

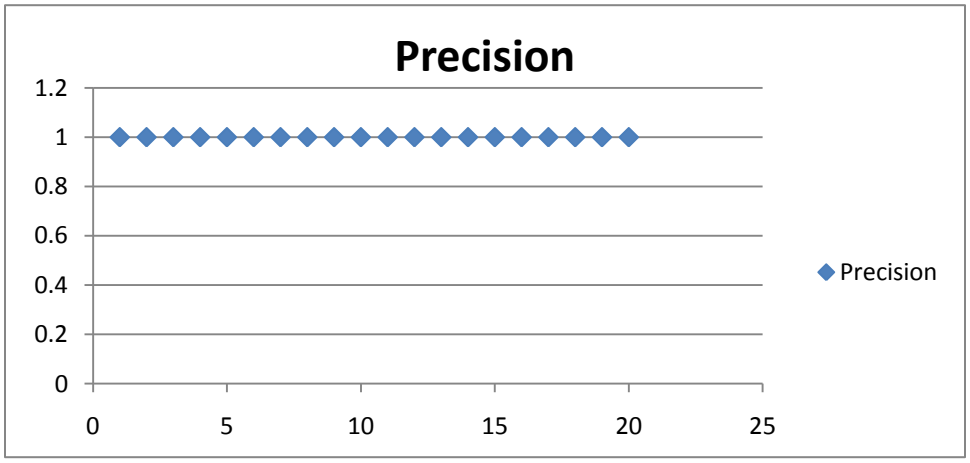

Figure 5.1 Precision graph of Query based search (applications)

The above graph shows average Precision 1

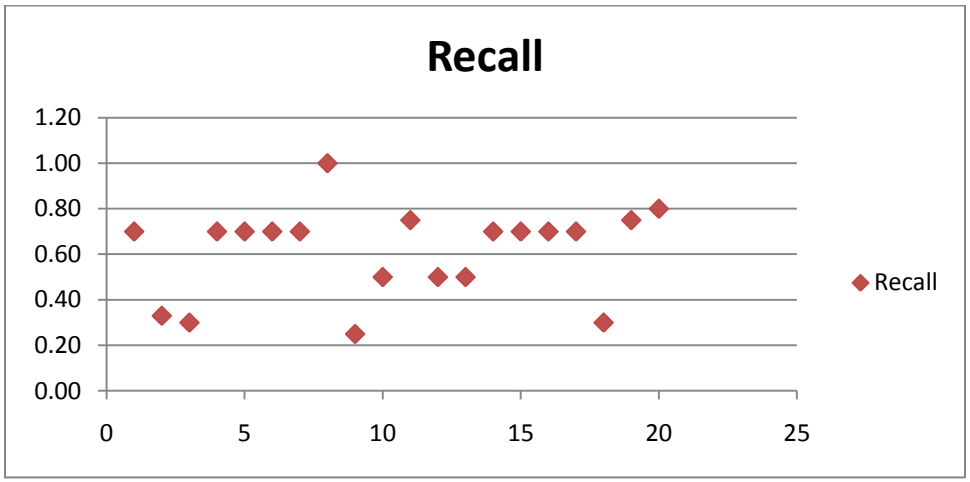

Figure 5.2: Recall graph of Query based search (applications)

The above graph shows average Recall 0.61

5.1.2 Keyword based search

In this search the user enters the keyword and based on this the result is displayed. User can enter any kind of keyword. If that keyword is present the user will get result otherwise the user gets result that "No such type of component is present in the repository".

Table 5.2: Keyword based search for various applications

Test cases	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Precision	0.75	0.66	0.66	0.8	0.8	0.8	0.75	0.9	0.75	0.75	0.8	0.75	0.9	0.75	0.8
Recall	0.66	0.5	0.75	1	0.75	0.66	0.66	0.8	1	0.75	0.5	0.8	0.75	0.66	0.9

Graphs of precision and Recall

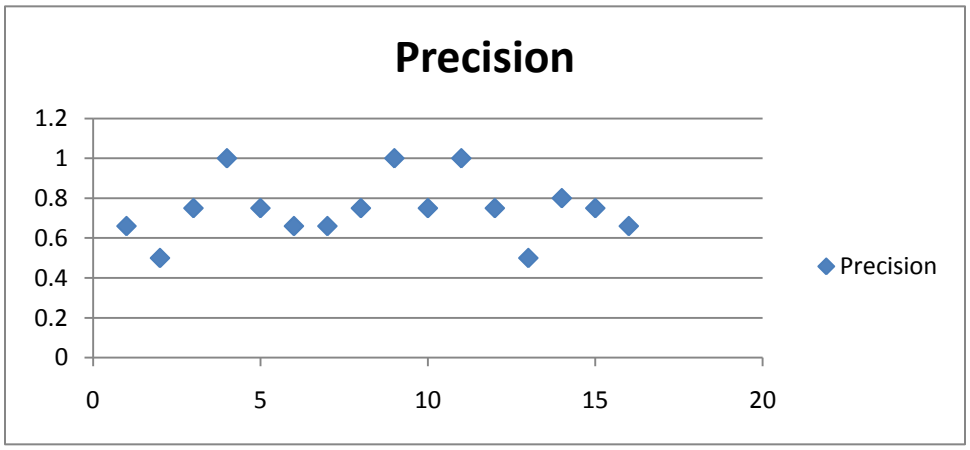

Figure 5.3: Precision graph of keyword based search (applications)

The above graph shows average Precision 0.77

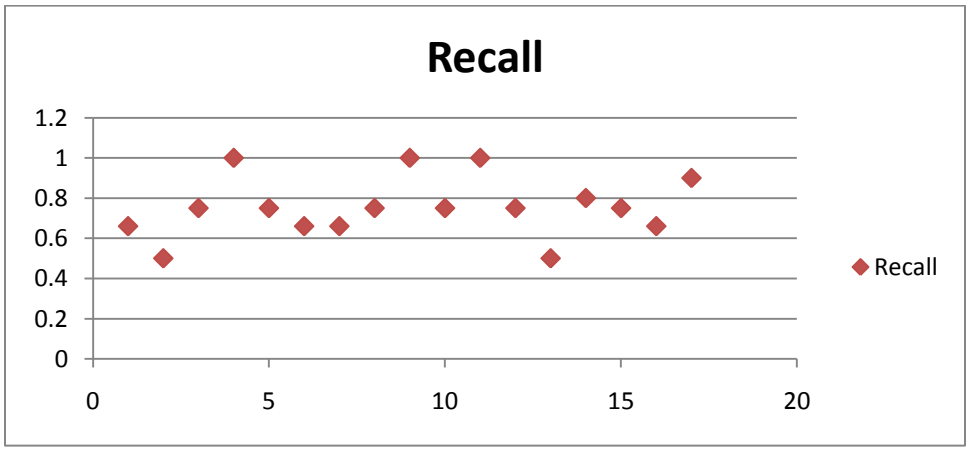

Figure 5.4: Recall graph of keyword based search (applications)

The above graph shows average Recall 0.73

2. Search for various games such as Free Billiards, Tic tac toe, Fun and bullets, Bubble match.

Table 5.3: Query based search for various games

Test										Precision			Recall		
Case	Cname	Brand	Vno	TOA	Ctype	OS	Lang	Cost	RC	RR	TR	P	RR	TR	R
1	Free Billiard	√	x	x	x	x	x	x	Mioplanet	1	1	1	1	2	0.5
2	Free Billiard	x	√	x	x	x	x	x	1.1	1	1	1	1	2	0.5
3	Free Billiard	√	√	x	x	x	x	x	Mioplanet 1.1	1	1	1	1	2	0.5
4	Free Billiard	x	x	√	x	x	x	x	Test Case	1	1	1	2	2	1
5	Free Billiard	x	x	x	√	x	x	x	Cots	2	2	1	2	2	1
6	Free Billiard	x	x	x	x	√	x	x	Linux	2	2	1	2	2	1
7	Free Billiard	x	x	x	x	x	√	x	C	2	2	1	2	2	1
8	Free Billiard	x	x	x	x	x	x	√	>50	2	2	1	2	2	1
9	Bubble match	√	x	x	√	x	x	x	Mioplanet Component	1	1	1	1	3	0.33
10	Bubble match	x	√	x	x	x	√	x	1.1 C++	1	1	1	2	3	0.66
11	Bubble match	x	x	√	x	√	x	x	Test Case Windows	1	1	1	1	3	0.33
12	Bubble match	x	x	x	√	x	√	x	Component C++	1	1	1	2	3	0.66
13	Bubble match	x	x	x	√	x	√	√	Component C++ <50	1	1	1	2	3	0.66
14	Bubble match	x	x	x	√	x	x	√	Component <50	2	2	1	2	3	0.66
15	Tic tac toe	√	x	x	x	x	x	x	Microhard	3	3	1	3	3	1
16	Tic tac toe	√	√	x	x	x	x	x	Microhard 1.3	1	1	1	2	3	0.66
17	Tic tac toe	x	x	x	√	√	x	x	Cots Windows	2	2	1	2	3	0.66
18	Fun and Bullet	√	x	x	x	x	x	√	Sify >50	2	2	1	2	3	0.66
19	Fun and Bullet	x	x	x	x	x	√	√	C++ >50	2	2	1	3	3	1
20	Fun and Bullet	x	x	x	x	x	x	√	>50	3	3	1	3	3	1
Average										1			0.739		

Graphs of precision and Recall

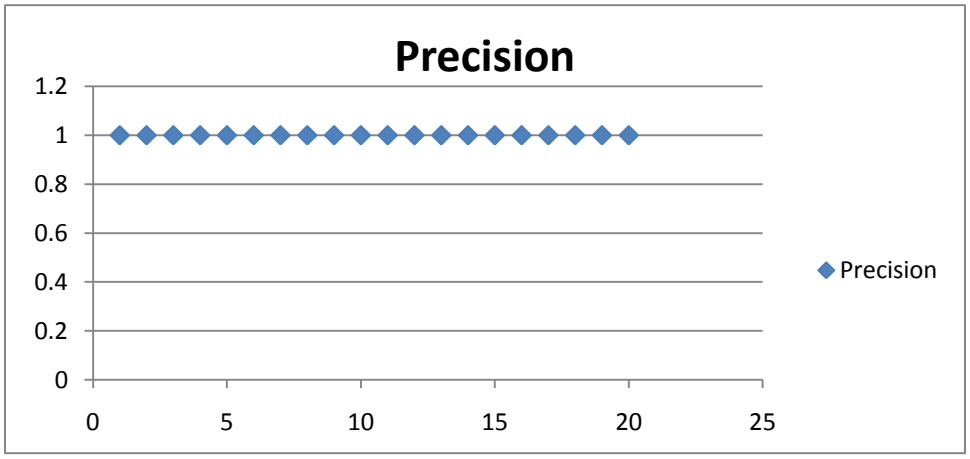

Figure 5.5: Precision graph of Query based search (games)

The above graph shows average Precision 1

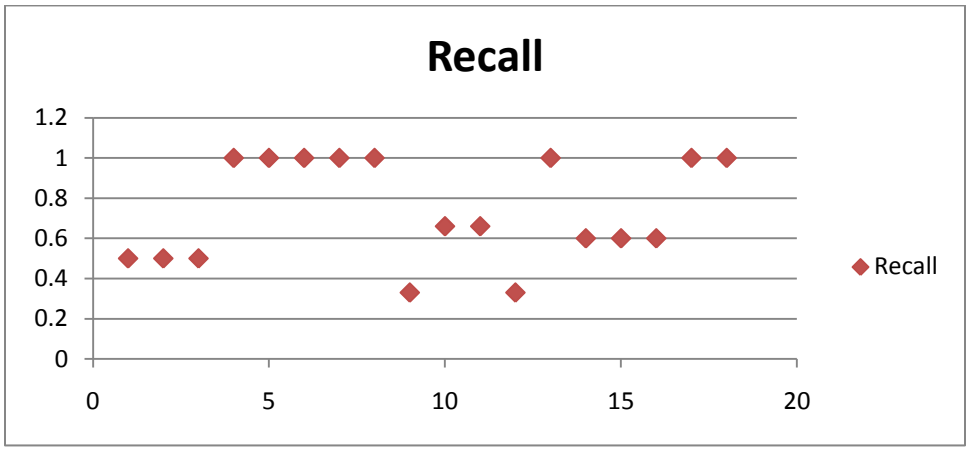

Figure 5.6: Recall graph of Query based search (games)

The above graph shows average Recall 0.74

Keyword based search for various games

Table 5.4: Keyword based search for various games

Test cases	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Precision	0.8	0.66	0.75	0.8	0.5	0.8	0.75	0.9	0.8	0.9	0.8	0.66	0.75	0.8	0.8
Recall	0.75	1	0.8	1	0.8	0.75	0.8	0.75	1	0.7	0.75	0.8	0.75	0.8	0.9

Graphs of precision and Recall

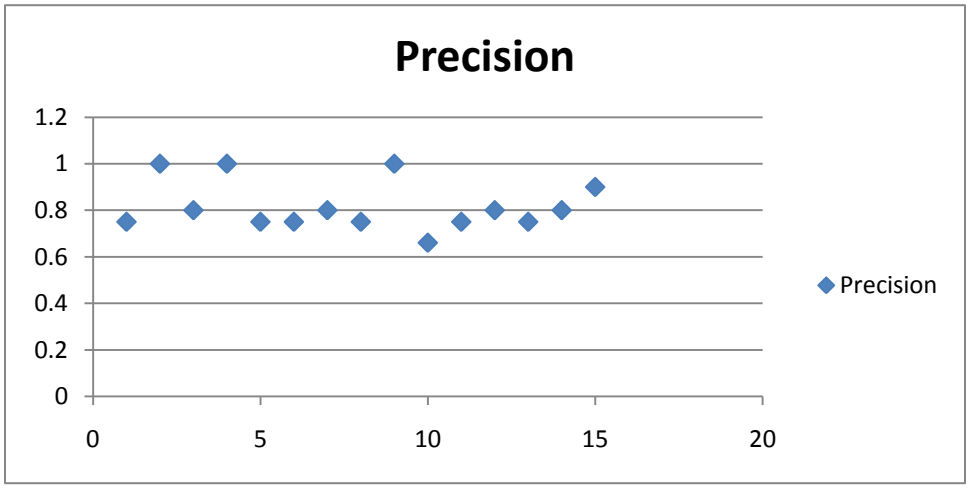

Figure 5.7: Precision graph of keyword based search (games)

The above graph shows average Precision 0.76

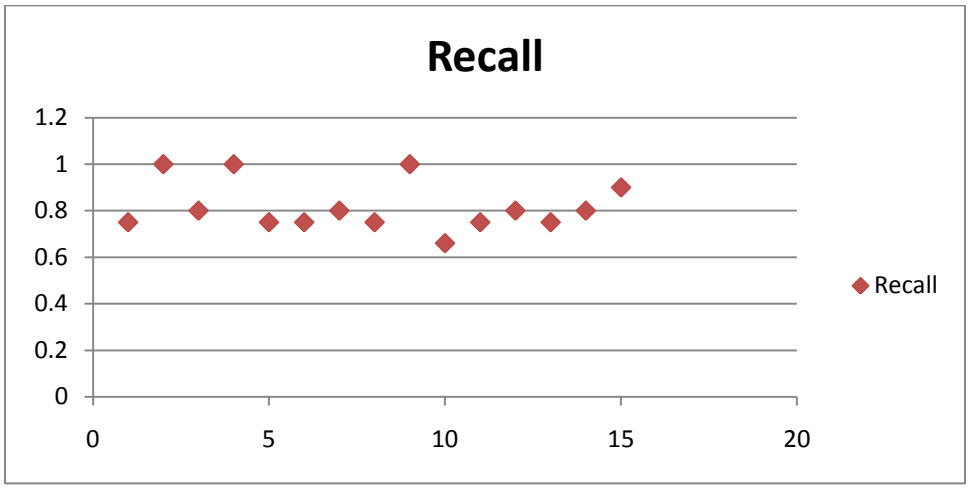

Figure 5.8: Recall graph of keyword based search (games)

The above graph shows average Recall 0.71

3. Search for various browsers such as Firefox, Google chrome, Safari installer, internet explorer.

Table 5.5: Query based search for various browsers

Test										Precision			Recall		
Case	Cname	Brand	Vno	TOA	Ctype	OS	Lang	Cost	RC	RR	TR	P	RR	TR	R
1	Firefox	√	X	X	X	X	X	X	Mozilla	3	3	1	3	3	1.00
2	Firefox	√	√	X	X	X	X	X	Mozilla 2.4	3	3	1	2	3	0.66
3	Firefox	X	X	√	X	X	X	X	Exe File	3	3	1	3	3	1.00
4	Firefox	X	X	X	√	X	X	X	Component	3	3	1	2	3	0.66
5	Firefox	X	X	X	X	√	X	X	Windows	3	3	1	2	3	0.66
6	google chrome	X	X	X	X	√	X	X	Linux	2	2	1	1	2	0.50
7	google chrome	X	X	X	X	X	√	X	C	2	2	1	1	2	0.50
8	google chrome	√	X	X	X	X	X	√	Google >50	2	2	1	2	2	1.00
9	google chrome	√	X	X	X	√	X	√	Google Linux >50	2	2	1	1	2	0.50
11	google chrome	X	X	X	X	√	X	√	Linux >50	2	2	1	1	2	0.50
12	Safari Installer	X	√	√	X	X	X	X	2.3 Exe File	3	3	1	2	3	0.66
13	Safari Installer	√	X	√	X	X	X	X	Apple Exe File	3	3	1	3	3	1
14	Safari Installer	X	X	√	X	X	X	X	Exe File	3	3	1	3	3	1
15	Safari Installer	X	X	√	√	X	X	X	Exe File Cots	3	3	1	2	3	0.7
16	Safari Installer	X	X	X	X	X	√	√	C >50	3	3	1	2	3	0.7
17	Internet Explorer	√	X	X	X	X	X	√	Microsoft >50	3	3	1	3	3	1
18	Internet Explorer	√	X	√	X	X	X	X	Microsoft Source Code	3	3	1	3	3	1
19	Internet Explorer	X	X	X	X	√	X	X	Windows	3	3	1	2	3	0.7
20	Internet Explorer	X	X	X	X	√	√	X	Windows C++	3	3	1	2	3	0.7
	Average									1			0.75		

Graphs of precision and Recall

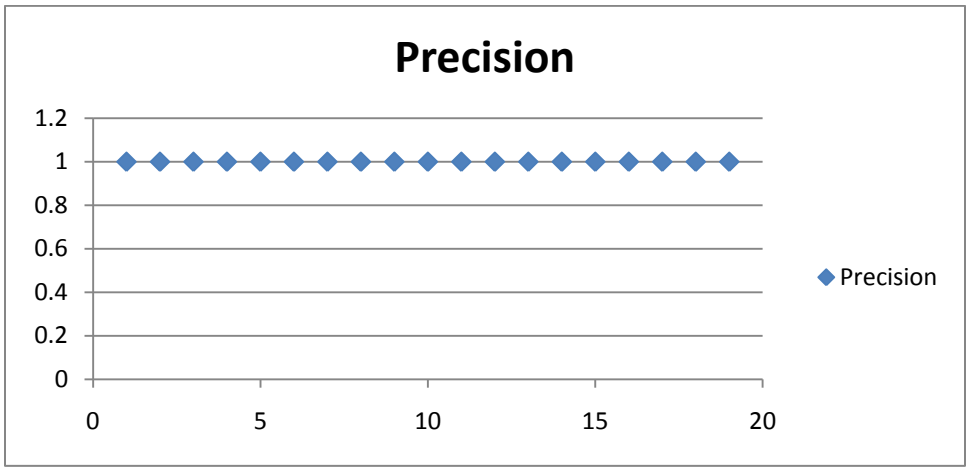

Figure 5.9: Precision graph of Query based search (browsers)

The above graph shows average Precision 1

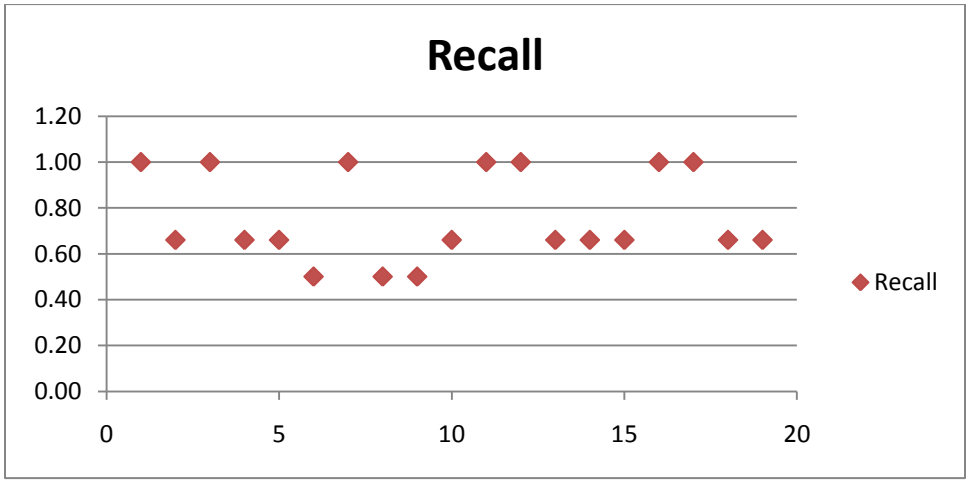

Figure 5.10: Precision graph of Query based search (browsers)

The above graph shows average Recall 0.75

Keyword based search for various browsers

Table 5.6: Keyword based search for various browsers

Test cases	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Precision	0.75	0.8	0.8	0.8	0.9	0.8	0.75	0.9	1	0.8	0.9	0.75	0.9	1	0.8
Recall	0.66	0.75	1	0.9	0.8	0.75	0.66	0.75	1	0.8	0.66	0.8	0.0.75	0.8	0.9

Graphs of precision and Recall

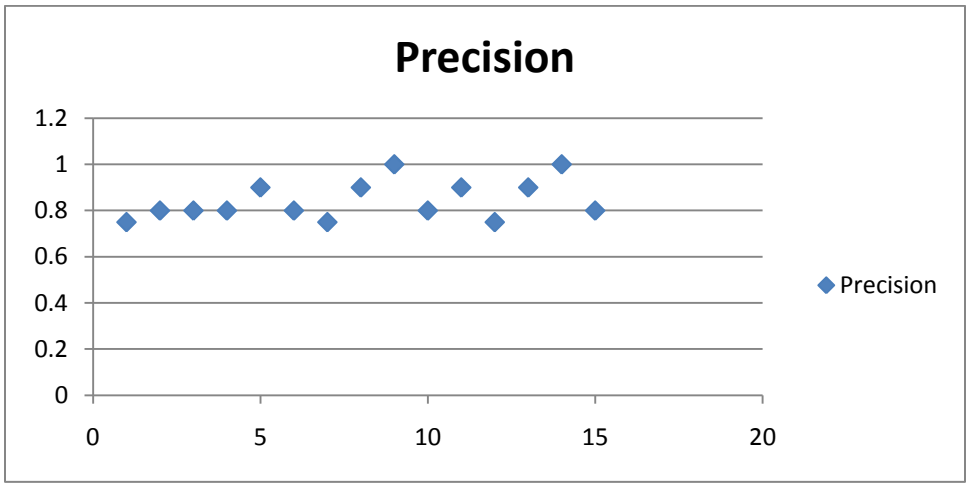

Figure 5.11: Precision graph of keyword based search (browsers)

The above Graph shows average Precision 0.84

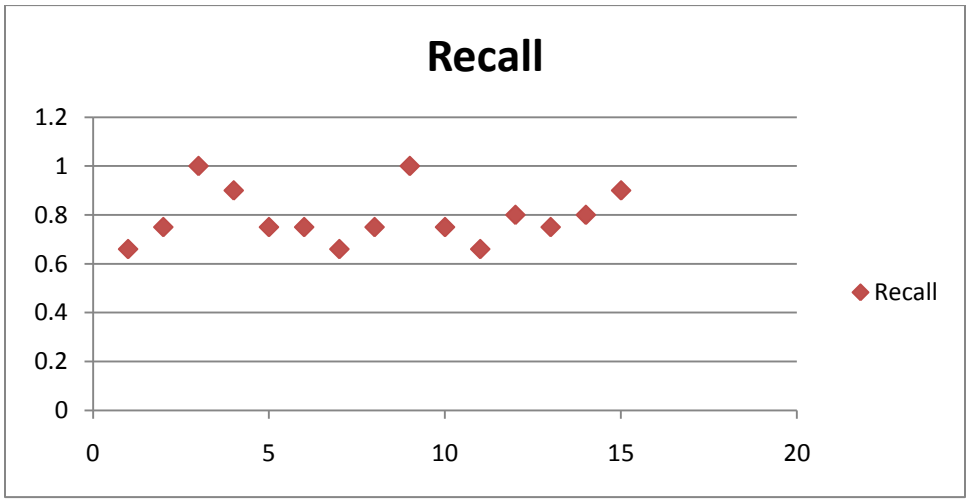

Figure 5.12: Precision graph of keyword based search (browsers)

The above graph shows average Recall 0.79

5.2 Comparison results of component repository system

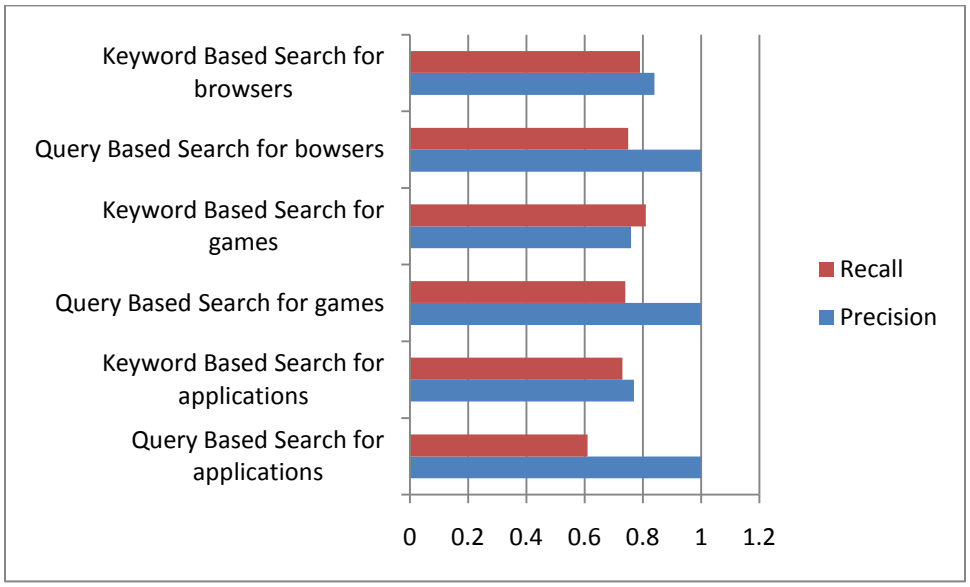

Figure 5.13: Comparison graph of two searches based on all three conditions.

Comparing the value of the precision and recall for the two searches for all the cases of applications, games and browsers we found that the keyword based search shows most encouraging results.

For instance, comparing the results of the Query based search in applications we have found that precision is 1 and recall is 0.61 and that of Keyword based search in applications the precision is 0.77 and recall is 0.73.

Now here the best recall seems that of Query based search, the value '1' depicts perfect recall but the fact that the number of components in the repository is according to multiple search criteria and the search input is able to retrieve all relevant components present.

Similarly comparing the values for other cases, we found Keyword based search gives the maximum value as they cover the greatest area in the repository as opposed to Query based search.

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

Chapter 6

Conclusion and Future Scope

This chapter discusses the conclusions of work presented in this thesis. The chapter ends with a discussion of the future direction which can be taken further.

6.1 Conclusions

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 with the help of multiple search criteria in the repository. An effective software tool with user friendly interface is designed and successfully implemented using proposed classification scheme. The system implemented also stores effectively the various reusable components in a manageable and understandable multiple categories of the search criteria from where users can extract the desired component effectively and efficiently. After implementation and various experimental results the following conclusions come to light:

- The implementation of the multiple search criteria in the component repository system makes the search more precise and the search results are more exact.
- Proposed classification scheme which combines the attribute value and faceted classification schemes for the multiple search criteria to classify components with the various attributes gives precise results.
- Out of all the comparisons of the search criteria the keyword based search is better as the measures of recall and precision are more accurate.

6.2 Future Scope

Future work involved with this classification scheme will be to refine the scheme for Multi-Tired or Multimedia presentation of components. Also various ways of optimizing system speed and efficiency must be explored in order to keep the repository as an effective design tool.

References

[1] J.C.Grundy, "Storage and retrieval of Software Components using Aspects", in Proc. of the 2000 Australasian Computer Science Conference, Canberra, Australia ,IEEE CS Press, pp. 95-103.

[2] P.C.Clements, "From Subroutines to Subsystems: Component-Based Software Development", American Programmer, Vol. 8, No. 11, November 1995.

[3] E.Smith, A.Al-Yasiri, M. Merabti, "A multitiered classification scheme for component retrieval", Proc. 24th Euro micro Conf, pp. 882–889, 1998.

[4] Ruben Prieto-Diaz Software Production Consortium, Herndon, VA, "Implementing faceted classification for software reuse", ACM Press New York, NY, USA, pp. 88 -97, 1991.

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

[6] H.Yao, L.Etzkorn, "Towards a semantic-based approach for software reusable component classification and retrieval". In Proceedings of the 42nd annual southeast regional conference, ACM Press, pp. 110–115, 2004.

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

[8] Mili, S.Yacoub, E.Addy, M.Hafedh, "Toward an Engineering Discipline of Software Reuse", IEEE Software Vol.16, No. 5, pp. 22–31, 1999.

[9] M.Gaedke, J.Rehse, G.Graef, "A Repository to facilitate Reuse in Component-Based Web Engineering", International Workshop on Web Engineering at the 8th International World-Wide Web Conference (WWW8). Toronto, Ontario.

[10] S.Shiva, L.Shala, "Using Semantic Wikis to Support Software Reuse", Journal of software, vol.3, No. 4, pp.1–8, April 2008.

[11] F.Siddiqui, "CBSE: A Look at Reusable Software Components", 1999.

[12] F.Church, "Software Reuse Executive Primer", Department of Defense, April 1996.

[13] P.Constantopoulos, M.Doerr, Y.Vassiliou, “Repositories for Software Reuse: The Software Information Base”, In Proceedings of the IFIP Conference in the Software Development Process, Como, Italy, 1993.

[14] W.B.Frakes, K.Kang, “Software reuse research: status and future”, IEEE Transactions on Software Eng 31, Vol. 7, pp. 529–536, 2005

[15] M. Hopfner, “Source Code Analysis Management and Visualization for PROLOG”, Dissertation for science doctoral degree, Julius, Maximilians, Universitat, Wurzburg, 2008.

[16] W.B. Frakes, T.P. Pole, “An Empirical Study of Representation Methods for Reusable Software Components”, IEEE Transactions on Software Engineering Vol.20, No.8, pp. 617–630, 1994.

[17] P.W.Foltz, W.Kintsch, “An Empirical Study of Retrieval by Reformulation on HELGON. Tech.”, Inst. of Cognitive Science, Univ. Of Colorado, Boulder, Colo, pp. 88-89, 1988.

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

[19] Ruben Prieto-Diaz, “Implementing Faceted Classification for Software Reuse” Software Production Consortium, ACM Press, pp. 88-97, 1991.

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

[21] Y.Ye, “Supporting Component-Based Software Development with Active Component Repository Systems” Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 2001.

[22] A.Mili, “Toward an Engineering Discipline of Software Reuse”, IEEE Software, Vol.16, No.5, pp. 22-31, 1999.

[23] R.Helm, Y.S.Maarek, “Integrating Information Retrieval and Domain Specific Approaches for Browsing and Retrieval in Object-Oriented Class Libraries” Proceedings of OPSLA'91, pp. 47-61, 1991

[24] A.Mili, R.Mili, R.Mittermeir, “Storing and Retrieving Software Components: A Refinement-Based System”, IEEE Transaction on Software Engineering, Vol. 23, No.7, pp. 445-460, 1997.

[25] T.R.G.Green, “Programming Languages as Information Structures, in Psychology of Programming”, J.-M. Hoc, et al., (eds.) Academic Press: New York, pp.118-137, 1990.

[26] N.Kaur, “Retrieving Best Component from Reusable Repository”, pp 23, 2005.

[27] Y.Ye, “Supporting Component-Based Software Development with Active Component Repository Systems”, PhD Thesis, University of Colorado, pp 87, 2001.

[28] C.B.S.Holanda, C.A.A.Souza, W.L.Melo, “ProReuso: A Web Component Repository Driven by a Software Reuse Process”, XV Brazilian Symposium on Engineering, pp. 208-223, 2001.

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

[30] S.Henninger, “ACM Transactions on software Engineering and Methodology”, Vol.6, pp. 116-120, April 1997.

[31] W.Frakes, T.Pole, " An empirical study of representation methods for reusable software components”, IEEE Trans. Softw. Eng.Vol. 20, No.8, pp 617–630, 1994.

[32] H.Mili, F.Mili, A.Mili, “Reusing software: Issues and directions”, IEEE Transactions on Software Engineering Vol. 21, No.6, pp. 528-561, June 1995.

[33] E.Motta, D.Fensel, M.Gaspari, R.Benjamins, “Specifications of Knowledge Components for Reuse”, In Proceedings of 11th International Conference on Software Engineering and Knowledge Engineering, Kaiserslautern, Germany, KSI Press, pp. 36-43, June 1999.

[34] H.H.Kagdi, M.L.Collard, and J.I.Maletic, “A survey and taxonomy of approaches for mining software repositories in the context of software evolution”, Journal of Software Maintenance, pp. 77-131, 2007.

List of Paper Published/Communicated

1. Vaneet Kaur, Shivani Goel, "Facets of Software Component Repository", International Journal on Computer Science and Engineering Vol. 3, No.6, pp. 2473-2476, June 2011 (Published)