International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 15 (2015) pp 35354-35361

Efficient Ranking and Retrieval of Reusable Software Components

Prof. Iqbaldeep Kaur[1]

[1]*Associate Professor, CSE Department*
Chandigarh University, Gharuan, Mohali
iqbaldeepkaur.cu@gmail.com

Satvir Kaur[2]

[2]*Research Scholar Chandigarh University, Gharuan, Mohali*
savimadahar29@gmail.com

Abstract
Component Based Software Engineering is gaining lot of importance in the field of software development. In component based development, the reusable components are embedded while not developing them from scratch. Retrieving components from software component repository for their efficient reuse is the challenging task. Component classification reduces the search space to search components. This reduces the search time for component retrieval. In this paper, we propose a new method for component classification and retrieval. Characteristics of uploaded components are extracted, and classification of components is performed the basis of their extracted characteristics. For retrieval of any component, it is searched by its required characteristics. The components having similar characteristics with the required one are resulted back which are further ranked using their download count and optimized using Ant Colony Optimization algorithm.

Keywords — Software reuse, Reusable components, Component classification, Component Retrieval, Ranking, Ant Colony Optimization.

Introduction

Component Base Software Engineering is a procedure that means to plan and develop programming frameworks utilizing reusable programming segments.
Programming reuse is one of the principle inspirations for CBSE. It concentrates on reusing and adjusting existing software components instead of creating them. This reduces both the development cost, effort and improves the quality of the software.
In CBSE the most challenging task is reusable software components storage and retrieval. Software component is any part of a software product that is independent and reusable. Software component can be any part of software development process i.e. specification, design, piece of code etc that can be used independently or can be embedded in any other software to provide specific functionality. Efficient storage and retrieval of components is very important because efficient storage leads to efficient retrieval.
To reuse any software component, user has to search it from software components available in the repository. User searches for the components that best matches with his requirement. There are various techniques for classification and retrieval of software components. Component classification techniques store the components in organized manner in the repository. Component classification reduces the search space by grouping together similar components i.e. components having similar characteristics.
In this paper we propose an efficient technique for classification and retrieval of components. In this technique, software components are classified by identifying their certain characteristics. During the component storage, certain characteristics of components are extracted by the proposed system. The classification of components is done on the basis of these characteristics (features) of the components. In the retrieval phase, the user searches the components according to his preferred characteristics which he wants the required component should possess. The user's preferences are passed into the proposed algorithm that returns those components whose characteristics match with the required characteristics. Further the resultant components are ranked. The ranked components can be further optimized using Ant Colony Optimization algorithm.
This paper is organized as follows: Section 2 discusses the existing component classification and retrieval techniques. In Section 3, we propose the proposed method for component classification and retrieval. In Section 4, implementation of the proposed system is shown. In section 5 and section 6, we discuss the experiments and results. Section 7 concludes this paper and discusses the future scope for this research.

Related Work

The most challenging task in component retrieval is to store the large collection of components in organized manner to result in fast and accurate retrieval. There are various techniques that are used for classification and retrieval of components. The most commonly used classification and retrieval technique is keyword based retrieval [13]. In this technique, reusable components are stored with the keywords i.e. components having similar keyword are grouped into one group. User searches component by keywords. All the

components whose stored keywords match with the user query are resulted back to the user.

Another technique for component classification is faceted classification [2].Each facet is the important characteristics of the component. Facet means assignment of component to multiple taxonomies i.e. set of attributes. Each facet makes classification of components in component repository from different perspective. Every facet has group of terms. The development of each facet is done by identifying important vocabulary in domain and grouping the relevant terms together into a facet. This method improves the search and retrieval of components and is also helpful in the development of a standard vocabulary for components and their attributes.

In enumerated classification [13], components are described by a defined hierarchical structure in which classes and varying levels of subclasses are defined. Actual components are at the leaf level. This type of classification can easily be understood by taking an example of library in which subject area like Biology, Chemistry etc. at the top level in hierarchy. Next level is sub-coded by specialist subject areas within main subject. These codes can again be sub-coded by author just like as in Dewey Decimal system [11].The main advantage of Enumerated classification is that it is easy to understand but the main shortcoming is its inflexibility. All the classes have to be initially defined. New components are inserted on lower levels. Enumerated classification though is fastest method but it is difficult to expand.

C.Srinivas, V.Radhakrishna (2013) proposes a new approach for clustering of reusable components or documents by using hybrid XOR function which finds the degree of similarity between any two components or documents. By applying hybrid XOR function a matrix named as similarity matrix is defined which is of order n-1 where n is number of components in a given set. The proposed algorithm takes the similarity matrix as input and gives output as set of clusters of components. The approach can be justified as it carries out very simple computational logic and efficient in terms of processing with reduced search space and can be also be used in general for document clustering or software component clustering [12].

Further C.Srinivad, V.Radhakrishna (2014) proposes another approach for clustering a given set of documents or software components by defining a similarity function called hybrid XNOR function to find degree of similarity between two document sets or software components. A similarity matrix is obtained for a given set of documents or components by applying hybrid XNOR function. We define and design the algorithm for component or document clustering which has the input as similarity matrix and output being set of clusters. The output is a set of highly cohesive pattern groups or components [15].

Semantic based retrieval technique [8][10] retrieves the components on the basis of description of the component and its domain. This technique retrieves components on the basis of domain or features of the components. Components are retrieved by matching with the meaning given for every component in the repository. Ontology based knowledge base stores the semantic information of the components.

In hypertext technique [3], the reusable components are organized in non-linear network of nodes and links. Hypertext technique is based on the concept of network of nodes and interconnection between them where each node is a reusable component or unit of textual information and interconnection between nodes are represented as links. The node from which the link starts is parent node and where the link ends is child node. Components can be searched by traversing the links.

In probabilistic technique [1], the components are retrieved according to their relevance to the user's requirement query. In all retrieval techniques, a document is retrieved when it has some sort of similarity with the query. It means that the documentation representation has certain probability of relevance to the query. When the similarity between the query and the documents representation increases, the probability of relevance also increases.

Proposed Methodology

This section presents a proposed methodology to efficiently classify and retrieve software reusable components. The first step is to classify software components. For this features of software components are extracted. In the second step the retrieved relevant software components are ranked and in the last step the ranked components are further optimized to get the most desirable component.

Following figure shows the block diagram of the proposed system.

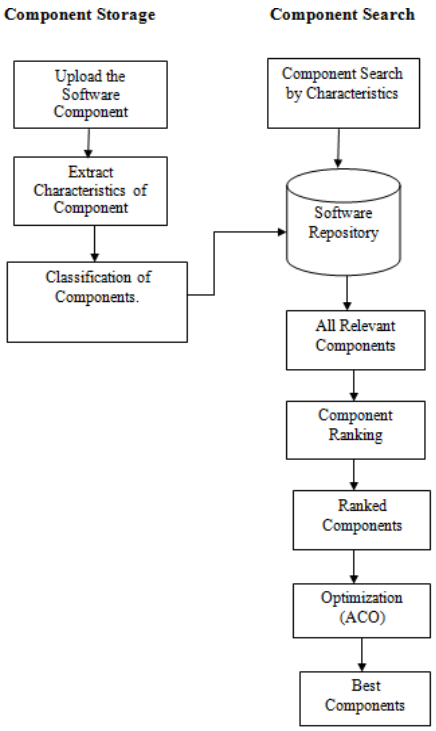

Fig 1. Proposed Methodology

In the proposed system, the reusable component is first uploaded. Then certain features of the uploaded component are extracted like language, processor, RAM utilized by the component etc.

Algorithmic Steps used for feature extraction of components are:

Steps:
1. Upload the component.
2. Apply String Tokenization to the content of component.
3. Now check the rule of various languages via string pattern matching functions in java based on which the system checks that this document belongs to which language, Operating System

A. Ranking of components:

Components that are retrieved on the basis of feature based search are ranked by the system according to their reuse frequency. An authorize user can login and search for his required component. Each time any authorized user searches and downloads any particular component, the rank of that component is increased by 1. Any component, which is downloaded more time, will have higher rank.

B. Optimization:

The components that are retrieved by feature based search are further optimized to get more accurate and optimized results. For this Ant Colony Optimization technique is used. Algorithm for Ant Colony Optimization is as follows:

```
procedure ACO_MetaHeuristic
While(not_termination)
generateSolutions ()
daemonActions ()
pheromoneUpdate ()
end while
end procedure
```

Implementation

Te proposed model consists of following modules.

I. User Registration/ Use Login
II. Upload Component
III. Search Component
IV. Rank Component
V. Optimization
VI. Download Component
VII. Graphical Results

In the first module user gets registered and login to the system. The authenticated user can add new component to the repository in the upload component module. While uploading the component, the user chooses the component to be uploaded and various features of that component like language, operating system, RAM utilization etc. are automatically extracted by the system and classifies that component according to its extracted features. The system returns success or failure of uploading the component. User can retrieve components in the search component module. In search module, the user first searches the component by his required features of the component. If no such component is there to which the required features match, the system returns message that no such document is found. Otherwise the system will retrieve all the relevant components that match with the user query.

Rank module provides rank to all the relevant retrieved components in the previous step. Rank is calculated for each retrieved component on the basis of its download count i.e. how many times that particular component is downloaded by that user in the past. Next module is optimization in which the ranked components are further optimized using Ant Colony Optimization algorithm. This optimization results into best components that best match with the user requirement. Next, the user can download his required component.

Experimentation

This section describes various experiments conducted in the proposed system.

A. Upload Component

In this experiment, the component is uploaded into the system by an authorized user. To upload the component, user chooses the component to be uploaded and the system automatically extracts characteristics i.e. feature of that component, classify it by its extracted features and stores into the repository. If the selected component is already stored into the repository, the system returns message that this component already exists.

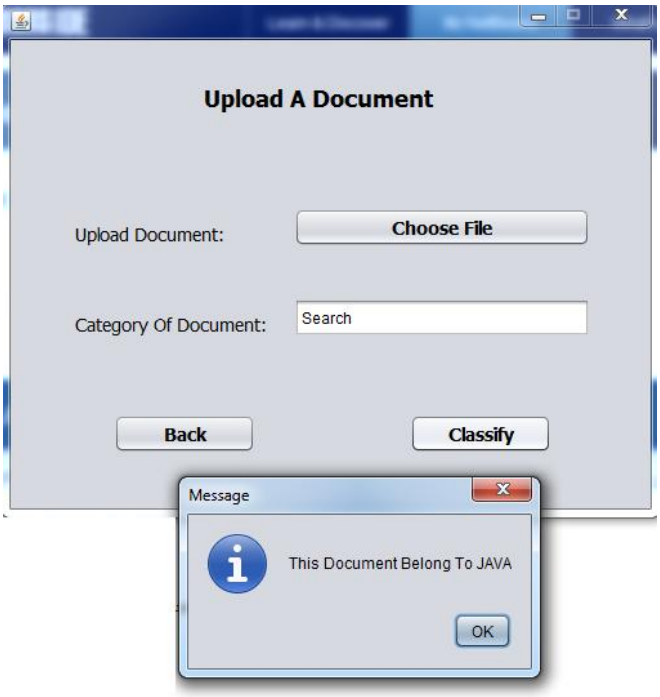

Fig 2. Upload Component

B. Search Component

This experiment is performed to search and retrieve the above uploaded components. Firstly the required features of component are entered by the user according to his choice.

These features are searched in the repository and all the relevant components are retrieved.
The user enters features namely:
Functionality- Search
Language- Java
Operating System- Windows, MAC, Linux
All the relevant components for this query are resulted back by the system.

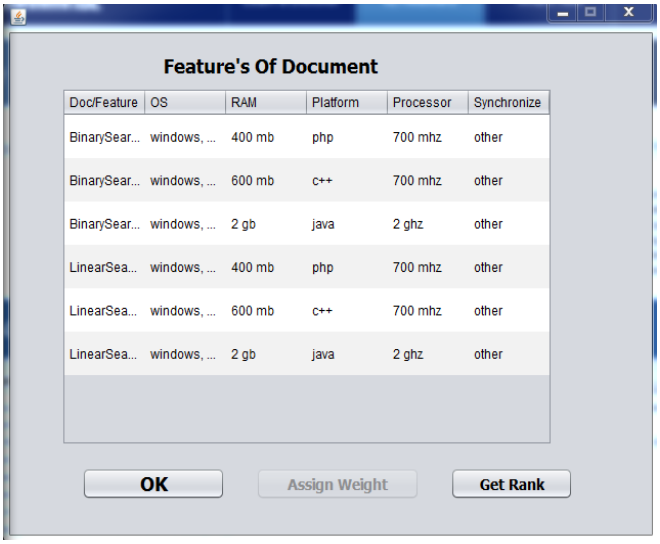

Fig 3. Feature Based Search

C. *Rank Component*

In this module, all relevant components that are retrieved in previous step are ranked by the system. The system provides rank to the component on the basis of their download count. The Component that has been downloaded more times, has the high rank.
The ranking of retrieved components is shown in following figure:

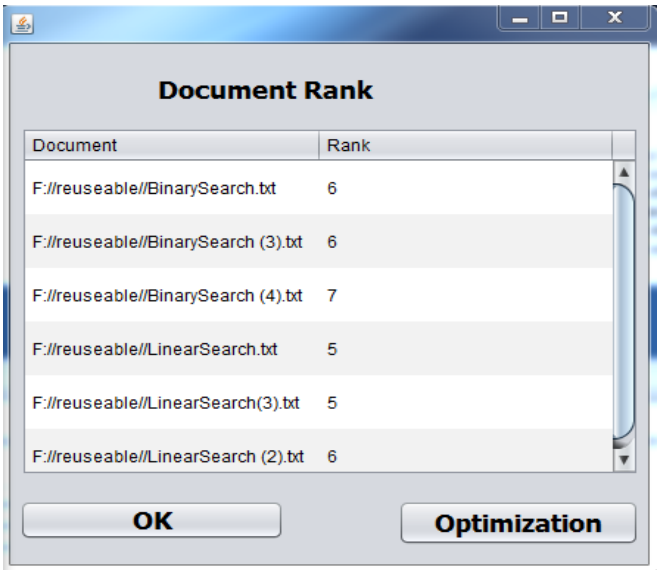

Fig 4. Ranking of Components

D. *Optimization*

To get more optimized results for user query, ACO is used. After the components are provided ranks, they can be optimized as:

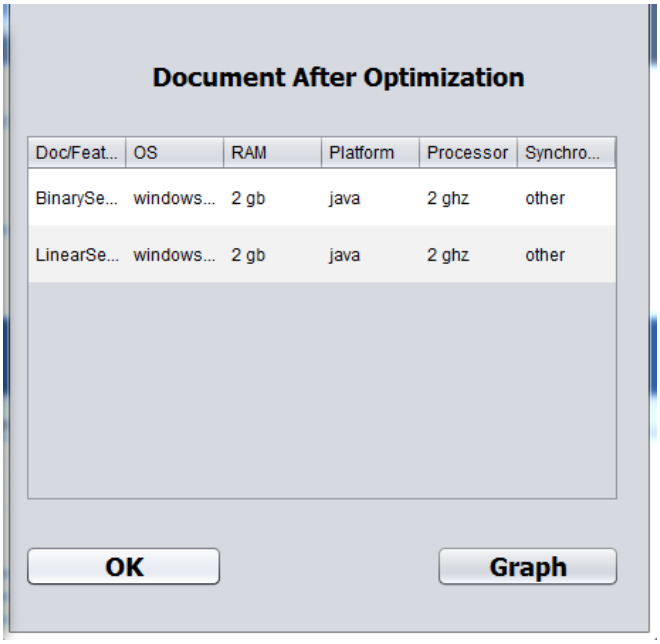

Fig 5. Optimized Results

E. *Download Component.*

The user can download his required component from the resultant components.

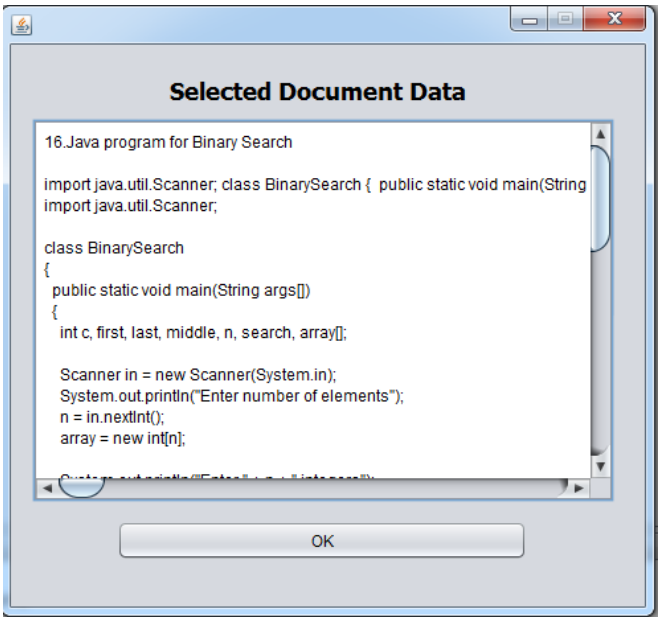

Fig 6. Download Component

Results and Analysis

This section discusses results of various experiments performed. Performance of the proposed system is evaluated using following parameters:

Precision: Precision is the ratio of relevant components that are retrieved by a query to the total number of components in the repository.

Precision=Number of Retrieved relevant Components / Total No. of Components in the repository.

Recall: Recall is the ratio of retrieved relevant components to the total number of relevant components in the repository.

Recall= Number of retrieved relevant Components / Total No. of Relevant Components

Efficiency: Efficiency of any retrieval system is determined by its ability to retrieve relevant software components from repository. Efficiency of the proposed system is calculated by following formula:

Efficiency= TC + (2 * P (R-P) / (R+P) *100)

TC: Total Components

P: Precision

R: Recall

Various experiments that are performed to evaluate the performance of proposed system and their corresponding results are shown in table1.

TABLE 1. Feature based Classification and Search

S. No	Features of Component	Relevant Components	Total Components	Precision	Recall	Efficiency
1.	Search	6	48	0.13	0.13	48
2.	Area	8	48	0. 17	0.17	48
3.	Sort	14	48	0.29	0.29	48
4.	Factorial	7	48	0.15	0.15	48

Table 1. shows various experiments that are performed to search components using feature based search and the precision, recall and efficiencies results of various experiments. In this table second column shows the features used by user to search components in different experiments.

Third column shows all the possible relevant components retrieved by the queries. Fourth columns shows total components stored in the repository. Fifth, sixth and seventh column shows the precision, recall and efficiency results respectively for each experiment.

After feature based search, experiments are performed for integrated classification and search i.e. optimized search. Following table shows results of integrated feature and optimized search technique.

TABLE 2. Integrated Classification and Search

S. No	Features	Retrieved Relevant Components	Total Relevant Components	Total Components	Precision	Recall	Efficiency
1.	Search, Java	2	6	48	0.04	0.33	54.27
2.	Area, Java	3	8	48	0.06	0.38	56.73
3.	Sort, php	5	14	48	0.10	0.36	59.3
4.	Factorial, C++	2	7	48	0.02	0.28	54

Table 2. shows results of integrated feature and optimized search technique. Second column shows the features used by user to search any component. When more features are used to search, the result will be more specific. Third column shows the relevant components to the query that are retrieved. Fourth and fifth columns show the total number of relevant components in the repository and all total components respectively. Sixth, seventh and 8th columns show the precision, recall, and efficiency results.

A. Graphical Comparison

Following graphs show comparison between precision, recall and efficiency results of feature based search and integrated search technique for all above experiments.

Precision Comparison Graph:

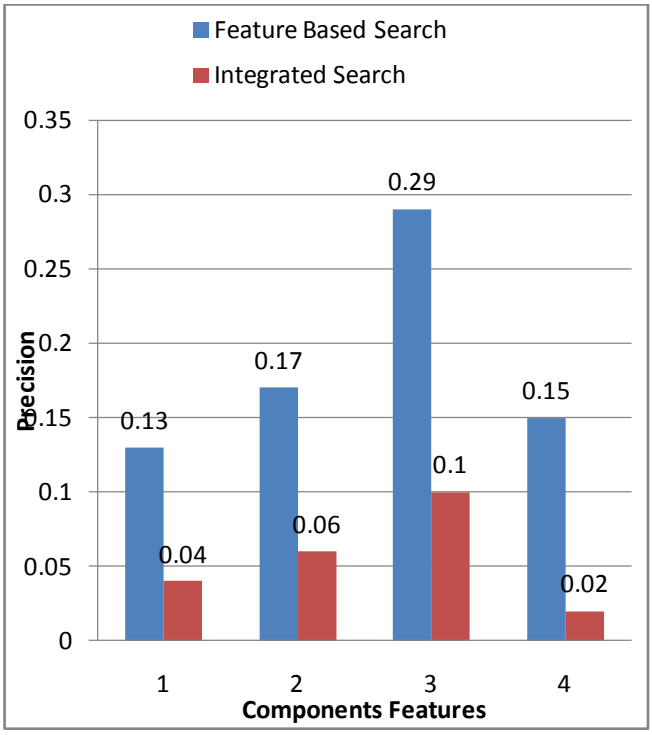

Fig 7. Precision Comparison

In above graph, precision results of both feature based search and integrated search techniques are shown. In this graph x-axis represents the features of components for both search techniques. Y-axis represents the precision results of various search operations performed. The values of precision of feature based search technique varies from 0.13 to 0.29 and the precision values for integrated search technique varies from 0.02 to 0.08.

i. Recall Comparison Graph:

In following graph, recall results of both feature based search and integrated search techniques are shown. In this graph x-axis represents the features of components for both search techniques. Y-axis represents the recall values of various search operations performed. The recall values of feature based search technique varies from 0.13 to 0.29 and the precision values for integrated search technique varies from 0.28 to 0.38.

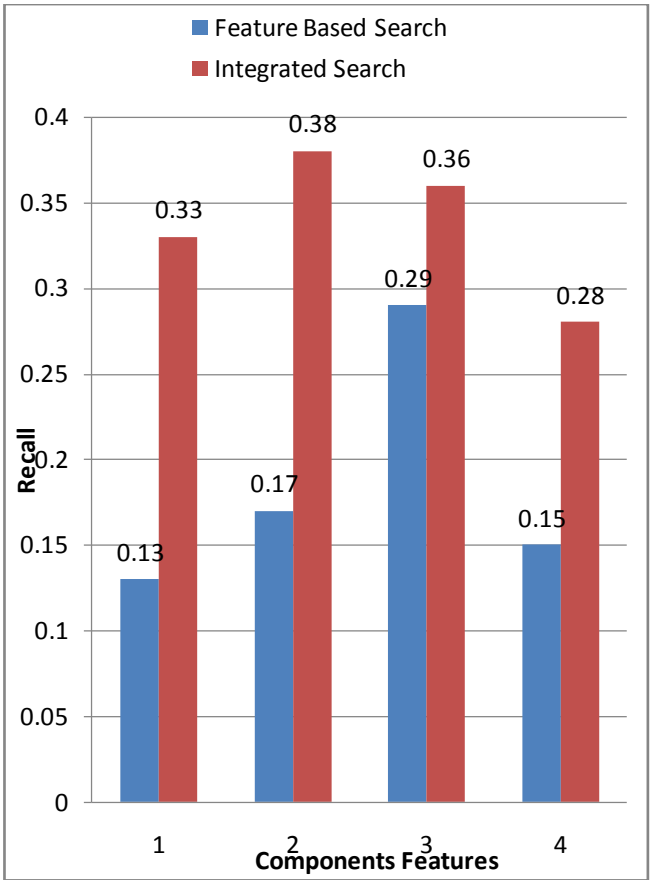

Fig 8. Recall Comparison

In the proposed system, the efficiency of component classification and retrieval is high. The integrated feature based and optimized technique results into higher efficiency than only feature or keyword based search. Following is the comparison graph of efficiencies of both the techniques.

ii. Efficiency Comparison Graph:

In the following graph, the efficiency results for both feature based technique and integrated technique are shown. In integrated technique efficiency is increased for each experiment performed as shown in graph.

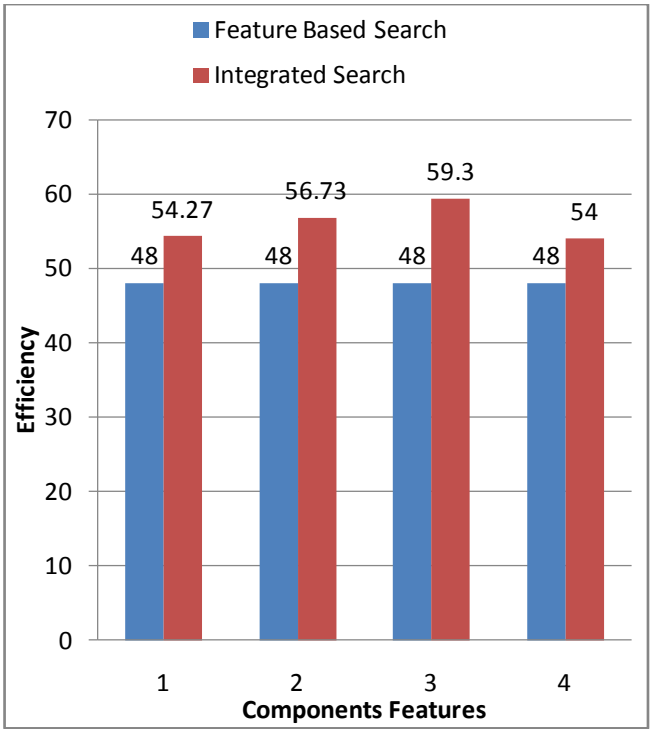

Fig 9. Efficiency Comparison

Conclusion and Future Scope

Effective storage and retrieval of reusable software components from repository is very challenging and time consuming task. The reusability of software component depends upon effective retrieval. In the absence of such retrieval mechanism, the significance of software reuse degrades. So, to improve the effectiveness of retrieval process, a hybrid approach is proposed in which there are feature based search, ranking mechanism and ant colony optimization techniques. Feature based search gives all the possible relevant components. Ranking algorithm is used to assign ranks to these resultant components on the basis of their download count. Finally, ACO is used to get optimal solution i.e. best components.

Possibilities for future work include:

- Any other ranking technique can be used to assign ranks to the components.
- Ant Colony Optimization technique can be replaced by or combined with other optimization techniques to get more optimized results.

References

[1] M.Gordon, "Probabilistic and genetic algorithms in document retrieval", *Communications of the ACM* 31, no. 10, pp.1208-1218, 1988.

[2] R. Prieto-Diaz, "Implementing faceted classification for software reuse", *Communications of the ACM* 34.5, pp.88-97, 1991.

[3] S. Jacques, and D. Desbois, "Information retrieval in hypertext systems: an approach using Bayesian networks", *Electronic publishing* 4, no. 2, pp. 87-108, 1991.

[4] V.Maxville, J. Armarego, and C. P. Lam, "Intelligent component selection," In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, IEEE, pp.244-249, 2004.

[5] N.Kaur, "Retrieving Best Components From Reusable Repository," Master Thesis, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, 2005.

[6] A. S. Andreou, D. G. Vogiatzis, and G. A. Papadopoulos, "Intelligent classification and retrieval of software components," In *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, IEEE, pp. 37-40, 2006.

[7] C.V.G Rao and P. Niranjan, "An integrated classification scheme for efficient retrieval of components," *Journal of Computer Science* 4.10, pp.821-825, 2008.

[8] C Li, X. Liu, and J. Kennedy, "Semantics-Based Component Repository: Current State of Arts and a Calculation Rating Factor-based Framework," In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pp. 751-756, IEEE, 2008.

[9] S. G. Khode, "Ant Colony Optimization Based Software Component Retrieval." PhD diss., THAPAR UNIVERSITY PATIALA, 2009.

[10] N. A. Saiyd, I. A. Said, and A. H. Takrori, "Semantic-Based Retrieving Model of Reuse Software Component," *IJCSNS* 10, pp. 154-161, 2010.

[11] N. Malik, "Proposed Classification Approach for Software Component Reuse," *International Journal of Electronics and Computer Science Engineering (IJECSE, ISSN: 2277-1956)*, pp.1993-1999, 2012.

[12] C. Srinivas, V. Radhakrishna, and C.V. G. Rao, "Clustering Software Components for Program Restructuring and Component Reuse Using Hybrid XOR Similarity Function," *AASRI Procedia*, vol.4, pp.391-328, 2013.

[13] A. Kumar, "Software Reuse Library Based Proposed Classification for Efficient Retrieval of Components," *International Journal of advanced research in computer science and software engineering," Vol 3*, pp.884-890, 2013.

[14] P. Niranjan and C.V.G Rao. "Dynamic grading of software reusable components for effective retrieval of components." In *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on*, IEEE, pp. 7-13. 2013.

[15] C. Srinivas, V. Radhakrishna, and C.V G. Rao, "Clustering software components for program restructuring and component reuse using hybrid XNOR similarity function," *Procedia Technology*, vol.12, pp.246-254, 2014.

[16] V. Verma, M. Mehta, and N. Adlakha, "Augmentation of Component Retrieval Using Ceremonial Methods," *International Journal of Research in engineering*, vol.4, pp.26-35, 2014.

[17] A.Singh, Janhavi, "Development of Component repository," *International Journal of Advanced research in Computer Science and Software Engineering*, vol.4, pp.952-957, 2014.

[18] K. Sandhu, and T. Gaba, "A Novel Technique for Components Retrieval from Repositories," *An International Journal of Advanced Computer Technology*, vol.3, pp.912-920, 2014.

This work was supported in part by the CSE Department of Computer science and engineering‖ , Head and faculties.

Prof. Iqbaldeep Kaur is Associate Professor of department of Computer Science and Engineering, University institute of Engineering, Chandigarh University Gharuan.(E-mail: iqbaldeepkaur.cu@gmail.com)

Er. Satvir Kaur is ME Student, department of Computer Science and Engineering, University institute of Engineering Chandigarh University Gharuan (E-mail: savimadahar29@gmail.com).