Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Reuse Libraries Based Proposed Classification for Efficient Retrieval of Components

Amit Kumar*
M.Tech., Department of Computer Science and Applications,
Kurukshetra University, Kurukshetra, Haryana, India.

Abstract*— Software Component Reuse has applied to a wide spectrum of software development. Software Component Reuse provide cost, time saving, etc. with increased product quality and decreased development cost. The component based approach involves constructing the application from different kinds of components store in reusable repositories. To retrieve the exact component from reusable repositories is a very important task. For reusing them, components are classified into different classes to store them in reusable repositories. This paper consider different classification schemes of components with a planned classification for efficient retrieval scheme.

Keywords*— Software reuse, Reuse libraries, Component, Component Classification, Component Insertion & Retrieval

I. INTRODUCTION

The demand for new software applications is currently growing, as is the cost to develop them. The number of qualified and experienced professionals required for application development is increasing with time (e.g. [1]). Most of them use reusable component to develop application rather than build from scratch. They constantly reused sections of code, templates, functions, and procedures because reuse always save time and money, and produce more stable and reliable product. Reusable software components have been promoted in recent years. Software reuse is to construct a new software by using existing software or software knowledge. Reusability of Components requires some kind of libraries where component can be stored, retrieved and modify easily. Storing, searching and retrieving components from a library of reusable components is main goal to reuse. This kind of library are also called Component Repositories(Libraries). Efficient reuse of a component means we can easily find out the component with knowledge of task it can performs and how we use it correctly. Component that are the candidate of reuse are executable objects, source code, link list, files, documentation, test cases, model projects, linkable libraries, idea's and testing strategies as shown in Fig. 1.

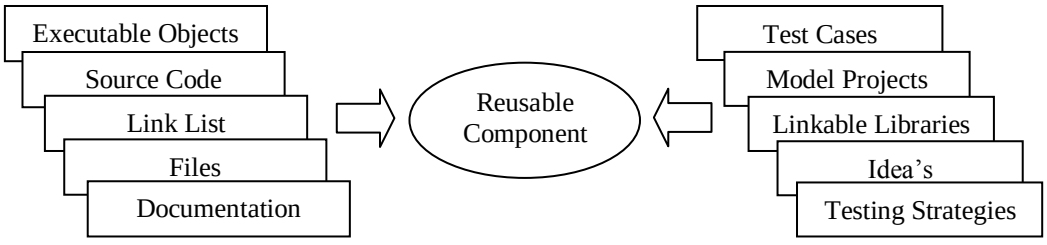

Fig. 1 Reusable Components

These components may be textual, graphical, audio or video, which requires multimedia presentation that will increase the user's understanding of the components(e.g. [1]). This increases the chance of retrieving the suitable components. On the other hand, as the information included in the library grows and varies the retrieval process becomes more complicated. This paper investigates component attribute with different types of classification schemes that enable components to be stored ready for retrieval. Finally looking at a possible way that each of these thoughts can be combined to provide a flexible classification and retrieval system.

II. RELATED WORK

Classification of reusable components has remained an important concept because software development process used reusable component. Software reuse was first envisioned by McIllroy (e.g. [2]), at a NATO Software Engineering Conference 1968, the main objective was to make something once and to reuse it several times. Different component classifications have come to make component reusable. Enumerated classification was based on the Dewey decimal system (e.g. [3]), in this hierarchical categories divided into subcategories, sub-subcategories, and so on. The advantage

of enumerated classifications is that they are easy to understand and use. A major shortcoming of the enumeration approach is its inflexibility. Disadvantages of enumerated classification have motivated the development of faceted classification, Prieto-Diaz (e.g. [4], [5]) focus is on organizing software collections for reuse using faceted classification of components. The conclusions of the experience was: reuse library technology is available, it is transferable, and it definitely has a positive financial impact on the organization implementing it. A new classification scheme called Attribute Value Classification was proposed by B. Burton et. al. (e.g. [6]), this classification uses a set of attributes to classify a component. Each attribute has the same weighting as the rest, so it is very difficult to determine how close the retrieved component is from require component. To further classify components Free Text classification was proposed by Frakes et. al. (e.g. [7]). Free text is the simplest form of identifying suitable components for reuse. Free text retrieval performs searches using the text contained within documents. The retrieval system is typically based upon a keyword search. It was not possible to reduce the query time. One approach to reducing the size of the indexed data is to use a signature matching technique (e.g. [8]). Based on signature match we cannot know which of large number of retrieved function does what. Liang Jun-Tao et. al. (e.g. [15]) presents a component classification model based on facet and neural network, with experiment. Xie Binhong et. al. (e.g. [16]) prove that component clustering algorithm based on the grade strategy makes the component clustering result more humanized and better serve user requirements. Thereby, the quality of component classification can be improved effectively. The component metrics can be considered and performance metrics for domain specific services can be identified as challenging research. N .Md lubair Basha et. al. (e.g. [17])

III. MANAGING COMPONENT

Component is the basic unit of software creation and it is independent part of software because it interacts with other components to accomplish a given task. Reusable software component are designed to take advantage in software development process. Every Component has its own attribute that it posses.

A. Components as reusable must have

1) *Understandability:* The component to be reused later by different people should be well documented and understandable. Imagine that you have to understand completely the components made by other and modify them to make it best for your system (e.g. [9]). Then you can not gain much from reusing them .So, user should have little effort spend for reading and understanding of components.

2) *Accessibility:* The component should be easy to get and reusable. Some time a person search for a component for a long time and he doesn't get it because that component is placed in a way that is not accessible easily. So, component should be placed in a way that is easy to find and reuse.

3) *Correctness:* The requirement for correctness becomes more important than ever, especially when we are reusing the program components made by other (e.g. [9]). Program Component should be correct as given in specification. Correctness has great impact on software reusability. Software Component may be use over and over again in the future. If is not correct than it is waste less and time consuming.

4) *Familiarity:* The component should be easily reusable by the people. They are more likely to use code that they already know how to use. Component should follow a general format with specification that easily understandable by people. If component language is not easily understandable than it is of no use for programmer.

5) *Adaptability:* The components to be reused need to be adaptable. This means that they are common so that they can be reused in as many alike systems as possible, and they require us less effort in altering and integrating them into a new system being built (e.g. [9]). This technique has been used for a long time after the use of subroutine in programming development.

6) Quality: Component should be in good quality for reuse. It should be well documented with great technology of design. It should work well according to its specification. Quality show the component behaviour. Some component may be written in simple language supporting small effort to specification.

Discovery and Retrieving software components can be prohibitively hard. For successful retrieval a meaningful organization of a collection of components is necessary. The better the organization is, the easier it is for reusers to find suitable components.

B. Component Classification Scheme

Classification is used to group alike components, i.e., all members of a group sharing one characteristic that components of other groups do not. Classification support to attach search information to components which be able to used for retrieval. Classification helps in identifying what measure can be used in matching process. A classification scheme for collections of reusable software components should meet the following measure:

- It must continuously hold important knowledge of different organisations.
- End User can easily update the repository.
- Component that are functionally equivalent must be easily findable.
- It must be in summarize and must have high expressive power .
- It must support to find similar component.
- It must be ready to support automation.

Several classification methods are applicable to software components, such as free text, keywords, facets and attribute-value pairs (e.g. [1], [10]).

1) *Free Text Keyword Classification:* Free text is the simplest type of identifying appropriate components for reuse. Apart from the exact documentation of any component designed for reuse, this approach requires no particular research for storage in the repository. Searching is complete text in nature. Attaching keywords to components is a different

means of detection. Keywords are entered by the developer of a component and its properties is also defined. Keywords can be chosen either freely or from a controlled terminology. For retrieval, users enter keywords which are compared to those of the components in the library. A typical example of free text retrieval is the 'grep' command used by the UNIX system. This type of classification generates large expenses in the time taken to index the material, and the time taken to generate a query. All the relevant text in each of the documents linking to the components are index, which must then be searched from beginning to end when a query is made. The major advantage of free text searching is that it can easily be fully automated. The disadvantages of free text classification is that it is confusing, involves indexing cost and nature of keywords is uncertain in nature. Another disadvantage is that a search result comes from many irrelevant components.

2) *Enumerated Classification:* Enumerated classification schemes are hierarchical categories separated into subcategories, sub-subcategories, and so on. The Dewey decimal system (e.g. 11]) is an example of an enumerated classification scheme used for the organization of book libraries. (e.g. Fig. 2) shows component of this classification. The advantage of enumerated classifications is that they are easy to understand and use by the people. A main shortcoming of the enumeration approach is its inflexibility. All categories have to be defined initially. New topics can simply be inserted at lower levels. The Enumerated classification though is fastest method but it is difficult to enlarge.

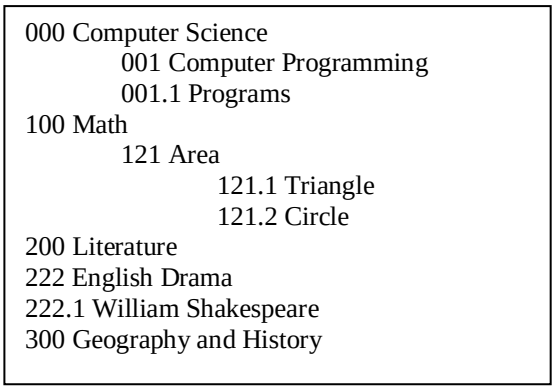

Fig. 2 Enumerated Classification

3) *Faceted Classification:* Faceted classification was proposed by Prieto-Diaz (e.g. [4], [5], [12]) also known as faceted navigation or faceted browsing that believe on facets which are extract by experts to describe the features about components ex:- Features, such as the component functionality, how to run the component, and implementation details. Facets are a normal way of organizing things. A domain area is analyzed and a set of basic features is identified. These features, named facets, are then prioritized by significance and connected to a component. A facet may be any of the following (e.g. 12]):

- Object(Array, Files, Character, etc.).
- Function(add, delete, move, compare, etc.)
- Medium(buffer, table, file, tree, etc.)
- Functional Area(bookkeeping, database management, etc.)

Type(file handler, lexical analyzer, scheduler, etc.) The benefit of faceted classification is flexibility. The facets and linked value can be added, deleted or modified simply. The problems with faceted classification is that the knowledge of domain specialist is implicit. It also provide difficulty in automation.

4) *Attribute Value Classification:* A set of attributes and values is designed for the attribute value classification. Attributes be able to take arbitrary values. Retrieving components requires the exact specification of values designed for attributes (e.g. [13]). Attribute value classification is similar to faceted classification with the following exception:

- Functional Area(bookkeeping, database management, etc.)
- There is no boundary on the number of attributes that can be used.
- Priorities are not assigned to Attributes.

For example, a book have many attributes such as the author, the publisher, its ISBN number and its classification code in the Dewey Decimal system. These are only example of the possible attributes. Depending upon who wants information about a book, the attributes could be concerned with the amount of pages, the size of the paper used, the type of print face, the publishing date, etc. This technique is the slowest technique with no ordering. Similar to enumerated classification, a shortcoming of attribute value classification is uncertainty. In different type of components different terms can be used to express the same values.

IV. COMPARISON STUDY OF CLASSIFICATION SCHEMES

Frakes and Pole have designed a prototype that will allow the user to select the type of classification scheme, and then use a dedicated user interface for that scheme (the presentation) (e.g. [14]). Using this prototype, Frakes and Pole conducted an study about the above classification methods (e.g. [14]). The investigation found no numerical evidence of any differences between the four different classification schemes, however, the following about each classification method was noted:

- *Enumerated classification:* Fastest method, difficult to expand.
- *Faceted classification:* Easily expandable, most flexible.

- *Free text classification:* Ambiguous, indexing costs.
- *Attribute value classification:* Slowest method, no ordering, number of attributes.

V. A PROPOSED CLASSIFICATION SCHEME FOR REUSABLE SOFTWARE COMPONENTS

Various Classification method has been proposed but their implementation and retrieval are different approaches. New component is added by the librarian using classification approach but during retrieval different approaches follow. As the number of components increases in the component database, the searching method become inefficient. One method of improving this situation is to classify the components with special techniques. Each of the four main classification schemes have equally advantages and disadvantages. The free text classification scheme does not provide the flexibility necessary for a classification system, and has too many problems with search spaces. The faceted system of classification provides the best flexible method of classifying components. However, it can cause problems when trying to classify very similar type of components for reuse in different domains. The enumerated system provides a quick way to search into a library, but does not provide the flexibility within the library to classify components for use in more than one way. The attribute value system allows multidimensional classification of the same component, but will not allow ordering of different attributes.

The new method that is proposed here is to apply the combination of classification method for classification of components. This has the characteristics of existing classification schemes. Here attribute value scheme is use to separate the libraries. These libraries then use a faceted and keyword based classification scheme to classify the components. Interface are provided to the users through which they know how to upload, retrieve and browse components. User will provide the information of the components to upload components. To retrieve the component user will give a query with detail and according to it a matching is done in repository and result is given. Then user can select his choice from similar component.

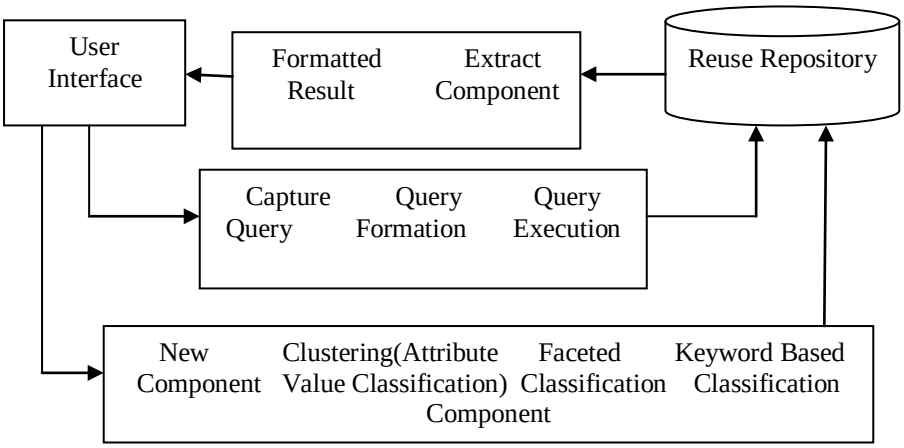

Fig. 3 The Architecture of a proposed Classification Scheme

In this method components from a common subject are grouped using enumeration classification in clusters. Each cluster being group of similar type of components. So first index is created to find the cluster for the type of component, that a person wants than index for components of every single cluster is created. So use of cluster approach initially shrink the search space. Now further keyword based classification method is applied to organize the component with in a single cluster.

A. Supporting Different Presentation

Different forms of presentation are stored in the library . This means that at any time, user can choose to view any form of presentation. Retrieving of one presentation format will not affect the choice of other forms of presentation. So, presentation of each component also classified using enumeration classification scheme to allow closer type of choices of presentation at any time. The administrator sets up the proposed system. The developers develop and put their components into library. Query tracking system should be maintained to improve the proposed system. The proposed system will provide the following functionality to users.

- Storing components
- Searching components
- Updating components

B. Supporting Different Version

Software component can generate many problem in the future. So, component develop today may have changes in the future. These changes may be error-correction (debugging), enhancements, or adaptation for a different use. Each of these changes may provide a new version of the same component. So, each version should be documented in the repository. During retrieving of component most relevant version should retrieved. Different people have different requirements from the same components. So, a version history also maintained in the repository. For this, component history is continuously updated. As the component changes, an addition is made to the component history. Even as a previous version of the component remains on the system, it will contained with in the history.

C. Insertion and Retrieval Algorithms

The algorithm to index the component repository is shown in (e.g. Fig. 4). Algorithm to retrieve the components is shown in (e.g. Fig. 5). Prototype Module of this system is shown in (e.g. Fig. 6.1, 6.2).

Algorithm: ***Component_Insertion***
/* **Input:** Component Facet and Attributes along with its documentation is given as input. Documentation includes brief description of the component. */
/* **Output:** This algorithm generates keywords using component descriptions for each component and insert component in appropriate repository. */

Begin

i. Input the description of the component with Facet and Attributes
ii. Process the description:
    a. Remove stop words.
    b. Stemming the words.
    c. Select Repository using Facet
iii. Insert the component in its selected repository at the end.

End.

Fig. 4 Component Insertion Algorithm

Algorithm: ***Component_Retrieval***
/* **Input:** Query in simple natural language with facet and attributes. */
/* **Output:** This algorithm retrieves a list of relevant components. */

Begin

i. Input the Query & Select Facet and Attribute.
ii. Preprocess the Query:
    a. Remove stop words.
    b. Stem the words.
iii. Select the repository based on Facet(Category).
1v. Repeat steps iv to vii
iv. For each Component Index in the repository, perform
v. ComponentRelevance factor = 0.
vi. Repeat step vii
vii. For each QueryKeyword in query index, perform
    a. Compare QueryKeyword with ComponentKeyword
    b. If match found
    ComponentRelevance factor = ComponentRelevance factor + 1.
    Until all Query_Keywords are exhausted.
Until all keywords in the component index are exhausted.
vii. Retrieve the components where ComponentRelevance factor > 0
ix. Sort the components on the basis of ComponentRelevance factor.
x. Display the components.

End.

Fig. 5 Component Retrieval Algorithm

VI. CONCLUSION

Various techniques have been evolved to classify reusable components. Each technique has been implemented differently. The proposed classification system gives idea to maintain repositories with better technology. It use different classification method for software components with their merits and flaws. It takes the advantage of the positive sides of each classification scheme. It explains that not every technique is appropriate for every type of component. In addition this classification method can perform both general as well as specific search. A capable software tool with user friendly interface is purposed with integrated classification scheme which restricts search space, reduces search time and increasing the efficiency of classification of software component.

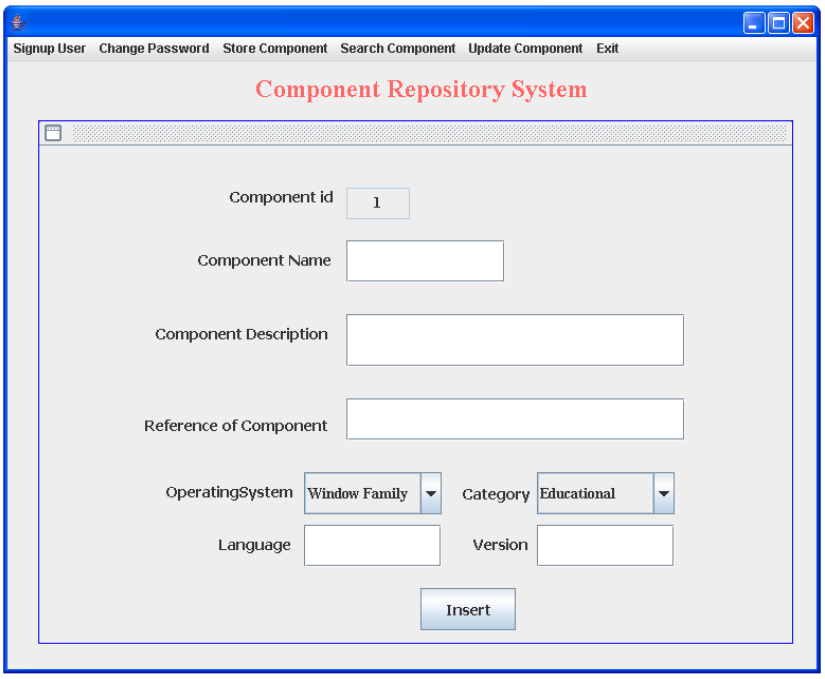

Fig. 6.1 Component Repository System Store Module

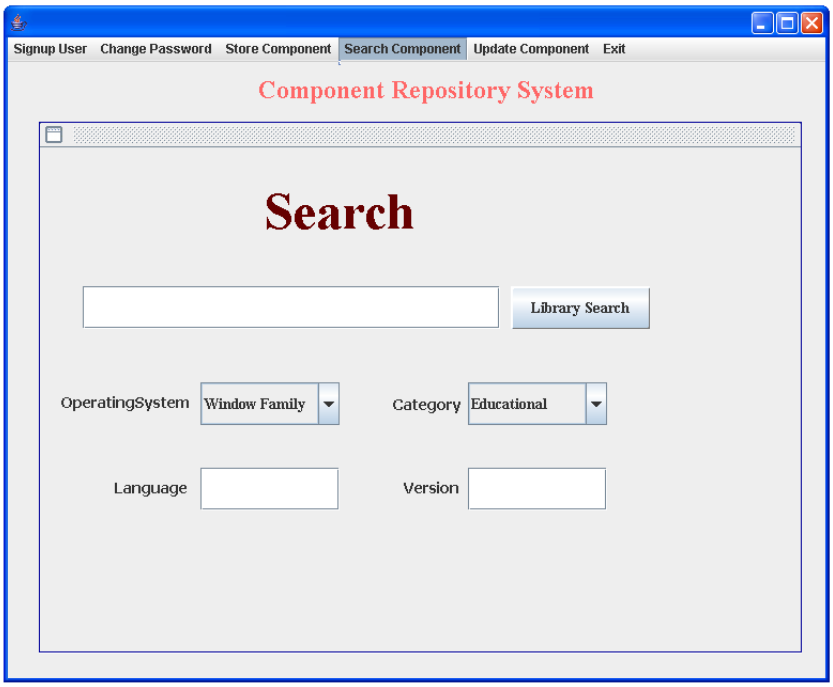

Fig. 6.2 Component Repository System Search Module

In Fig. 6.1 we can see how to add component in repository, we have to fill all detail required in that module to insert component. In Fig. 6.2 we can search components, we can search with query by selecting a facet or we can add attribute with query.

ACKNOWLEDGEMENT

Author is especially thankful to Dr. Bharat Bhushan, Associate Professor & Head, Department of Computer Science & Applications, Guru Nanak Khalsa College, Yamuna Nagar, Haryana, India for his valuable suggestions about this topic.

REFERENCES

[1] Ewan Smith, Adil Al-Yasiri and Madjid Merabti, *A Multi-Tiered Classification Scheme for Component Retrieval*, IEEE Software, 1998

[2] McIlroy, M.D. *Mass-produced Software Components. In Software Eng. Concepts and Techniques*, 1968 NATO Conf. Software Eng., ed. J.M. Buxton, P. Naur, and B. Randell, pp 88-98

[3] M. Dewey., *Decimal Classification and Relative Index*. Forest Press Inc., 19th edition, 1979.

[4] R. Prieto-Diaz and P. Freeman. , *Classifying Software for Reusability*. IEEE Software, January 1987, pp. 6-16.
[5] Rubh Prieto-Diaz, *Implementing Faceted Classification for Software Reuse*, Experience Report IEEE 1990
[6] B. Burton, R. Aragon, S. Bailey, K. Koehler and L. Mayes. *The Reusable Software Library*. IEEE Software, July 1987, pp. 129-137
[7] W. Frakes and R. Baeza-Yates, Eds., *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[8] Y. Maarek. Introduction *to Information Retrieval for Software Reuse*. In Advances in Software Engineering and Knowledge Engineering, Vol2, edited by: B. Abriola and G. Tortor. World Scientific Publications, Singapore, 1993.
[9] Victor R. Basili, John Bailey, Bok Gyu Joo, H. Dister Rombach, *Software Reuse: A Framework for Research*, National Aeronautics and Space Administration under Grant NSG-5123, by VITRO, and by Pena Central, 2001
[10] C.V. Guru Rao, P. Niranjan, *An Integrated Classification Scheme for Efficient Retrieval of Components*, Department of Computer Science and Engineering, Kakatiya Institute of Technology and Science, Warangal, Andhra Pradesh, 2008.
[11] M. Dewey., *Decimal Classification and Relative Index*, Forest Press Inc., 19th edition, 1979.
[12] Ruben Prieto-Diaz, Freeman , *Towards A Classification Model for Component Based Software Engineering Research*, Proceeding of the of the 29th EUROMICRO Conference © 2003 IEEE.
[13] Jeffrey S. Poulin and Kathryn P. Yglesias, "*Experiences with a faceted classification scheme in a large reusable software library (RSL)*", In Seventeenth Annual International Computer Software and Applications Conference, Phoenix, AZ, November 1993.
[14] W. Frakes and T. Pole. *An Empirical Study of Representation Methods for Reusable Software Components*. IEEE Transactions on Software Engineering, August 1994, pp. 617- 630.
[15] Liang Jun-Tao, Jiang Xiao-Yuan. A *Software Component Classification Based on Facet and Neural Network*. Second International Symposium on Intelligent Information Technology Application, IEEE 2008
[16] Xie Binhong, Ren Yaopeng, Zhang Yingjun ,Chen Lichao. *Research on the Clustering Algorithm of Component Based on the Grade Strategy*, 2010 International Conference on Computer Application and System Modeling, IEEE 2010
[17] Porter, M. F. An Algorithm for Suffix Stripping, Program, Vol. 14 No. 3, pp 130-137, 1980.
[18] N .Md lubair Basha, Dr. Salman Abdul Moiz. *Component Based Software Development: A State of Art*, IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012) IEEE 2012

