ISSN:2320-0790

An International Journal of Advanced Computer Technology

A Novel Technique for Components Retrieval from Repositories

Kanwaljeet Sandhu, Trilok Gaba
M.D University, Rohtak

Abstract: There are various models has been developed to describe the reusable components. These models defines what features should be represented with the components. Some typical component models are: 3C model, COM/DCOM, CORBA. Whereas a component is a unit of code that on execution performs some functions. A repository is used to store different types of components. So there is a requirement of identification of effective component search over the system model. A fundamental problem in software reuse is how to organize collection of components so that search and retrieval process become effective. Representations and retrieval techniques are interrelated. Software component comes under the information retrieval on which the effective component access over the software system. In this paper a suffix tree based component representation is defined in the form of a tree model. It is a structural model for information generation and representation so that the effective component modeling and component specification will be obtained. It is considered as the edge based modeling under the single character search. A suffix tree is build with various levels. With each level the pruning over the keyword list is performed. The efficiency of the system is based on the pattern specification so that the requirement analysis is defined.

1. Introduction

The component based software system basically derive the information from existing components or systems so that the software quality improvement along with productivity analysis is achieved. Software system design includes the development from some existing system so that the reliable software system will be improved [5]. These kind of software system are based on parallel processing in which the information extraction from existing work as well as the new software system will be generated partly. The component analysis is performed so that the reliable software system will be developed. Software system under boundary analysis is defined to improve the development process under component based modeling. The basic components associated with this system are:

1.1 Qualified Components-

These kind of software systems are used by software engineer for the performance analysis and reusability analysis. These components are incorporated under the quality analysis. The functionality of these components is fully compatible to the requirement so that the reusability ratio is higher.

1.2 Adapted Components-

These kind of components are defined along with some known and unknown characteristics analysis so that the development process will be improved.

1.3 Assembled Components

This kind of software system includes the architecture level analysis and the interconnection between the components is performed to coordinate the software component system.

1.4 Updated Components

This kind of software system includes the existing software system analysis along with version update.

A software component is defined as an independent object or the characteristic object that can be deployed or integrated in an application so that the development process will be improved. The description based component systems are defined to improve the component strength

- Component- is represented as an independent entity to improve the system architecture so that the object level improved will be obtained.

- Software component- defined the composition at the dependency level so that system improved will be achieved.
- Run-time software component- includes the component interface based development model
- Business component- is defined as the process model so that the effective development will be obtained.

Properties of Components

From the above discussion some generic properties about components can be concluded which are essential for describing a component those are independency, modularity and reusing abstraction.

- Independence- components are the individual modules that can be integrated to any application based on requirement analysis. The component properties are different form one other.
- Modularity- The module level analysis is defined under the functional abstraction so that component reuse will be obtained and the encapsulated development model is attained. The internal structure analysis is defined for component analysis. The interface analysis is performed for development procedure improvement.
- Reuse Abstraction- To rule the accessibility degree to the internals, a software component is associated to an abstraction when it is reused.
- Marketable Entity- A component must be robust so that the effective integration will be formed. Such component must be available in open market.
- Incomplete Application- components are capable to design the software system under domain task analysis.
-

In component reuse, the software systems are developed by using the existing code or the components. A software product includes the number of integrated components via some process on them. These code or components can be some language library code or some existing external code. To get the relevant components from the library is difficult task. It is required to know the architecture of the existing system as well as the architecture of the new developing system. If the architecture of both the software are similar then there is need to retrieve the component information from the software repository. Retrieving the information from the repository is difficult task because there are many components exist having the same name or tag. When the search is applied on the repository it returns the multiple and ambiguous results. There is need to arrange the results in proper way so that the user can get the required component. Hence there is need to develop a new technique which can help to solve the problem discussed above.

In this paper a new technique using the suffix tree to search the relevant component from the result is proposed. To get the components or code form the library, it is required to know architecture of the software well. Because the architecture can tell about class diagram, object diagram, and process diagram etc. But to build such architectural overview the first requirement is to retrieve the component information from the software product. An approach is presented to retrieve such information from the software product. This information is collectively called the Software Ontology. Once the components are retrieved they are maintained in the form of suffix tree. The suffix tree is way to maintain the component database in such way the component search can be performed in an easy and efficient way. The presented approach is capable to find the existence of any software component in software product. The main motivation of our research is to study different existing component searching techniques. Various searching techniques have been developed to gather and organize the components. But still retrieving relevant component from a large collection of components is a challenging task.

3. Related Work

Many reserachers have put efforts in this direction. Ning [36] presented a number of research efforts related to CBSE in his paper entitled "Component-Based Software Engineering (CBSE)". Author defined an effective component search mechanism to improve the software system under the component analysis identification. Author defined the basic model for component level definition for component search specification. Standish and Thomas [37] presented a paper on, "An Essay on Software Reuse". This paper explored basic software reuse concept and discussed briefly what economic incentives were used for software system generation so that the software reusability will be improved. The software system under the module definition is given to achieve the technical implications. Paper defined the component reuse under effective system software evaluation and development.

Chang *et al.* [38] presented a paper entitled "A Formal Approach to Software Components Classification and Retrieval". In this paper authors

performed a work on classification and retrieval of software components using formal approach. They proposed an approach in which the software system definition is performed under keyword specification and analysis. The component system integrated in such system is performed under functionality definition and analysis. Component specification based analysis is obtained under different component generation and specification so that the system level improvement will be obtained. Many researchers have done a lot of work in field of retrieving of software components from the past fifteen years. From these researches one thing is come into view that for effective reuse of any software component from repository, retrieval mechanism is very important. To enhance this retrieval an automation retrieval of software components was proposed by Luqi and Guo [39] through their paper "Toward Automated Retrieval for a Software Component Repository". The paper discussed the improvement of over existing software component system using signature matching and provide provide profile based match for effective development to the system is obtained.

Lucredio *et al.* [40] presented a paper, "Component Retrieval Using Metrix Indexing". If the software repositories have a large number of stored components, then efficient retrieval of these software components from the development modeling to the system. To make the retrieval process more efficient, the scenario specific search mechanisms are defined. Author defined the effective component utilization and component search under component search and retrieval. Garcia *et al.* [41] Presented a paper entitled "From Specification to Experimentation: A Software Component Search Engine Architecture". This paper includes the component specific search mechanism under the text based mining and component level identification. The methods are defined to improve the search process and search directions. This kind of software system models are defined under experimental analysis and development with effective research modeling and construction.

The organization of software repositories is yet again important aspect for effective component retrieval. Veras and Silvio [42] used clustering technique for the organization of software repositories and presented a paper "Comparative Study of Clustering Techniques for the Organization of Software Repositories". The paper discussed how software reuse was essential for improving the productivity and quality of software projects. They suggested that one of the key issues to promote the adoption of software reuse in companies was the development of effective repositories of software components. In this paper they quantitatively compared two clustering techniques, namely, self-organizing maps (SOM) and growing hierarchical SOM (GHSOM) for clustering a repository of classes from a Java API for building mobile systems. The simulations have shown that GHSOM outperforms SOM in these tasks. They resulted that GHSOM was more suitable for this task because it was a constructive technique.

Dixit and Saxena [43] performed a work on software component retrieval using the concept of genetic algorithm and presented their work in paper entitled "Software Component Retrieval Using Genetic Algorithms". Their proposed paper was an attempt on major issue of component based software engineering concerned that is Component Selection. The paper described how a Genetic Algorithms based approach can be used for component selection to minimize the gap between components needed and components available. Now a relevant objective has at hand in this direction that is to make use of these methodologies acceptable from the software engineering community. Therefore, in this paper Genetic Algorithms based approach for selection component has been developed. Another paper was presented by Ichii *et al.* [44] on "Software Component Recommendation Using Collaborative Filtering". The author premised through this paper that software component retrieval systems were widely used to retrieve reusable software components. This paper proposed recommendation system integrated into software component retrieval system based on collaborative filtering. This system used browsing history to recommend relevant components to users. A case study using programming tasks has been conducted and found that the proposed system enables users to efficiently retrieve reusable components.

Viana *et al.* [45] presented a paper entitled "A Search Service for Software Components based on Semi-Structured Data Representation Model". This paper presented the architecture, functionalities and implementation of a search service that adopts techniques for indexing semi structured data. The search service proposed in this paper performed the indexing of assets described using a semi structured data representation model, as opposed to automatic extraction of information from the source code or textual documentation approaches. Li and Luo [46] performed a work on "Component Retrieval Based on Domain Ontology and User Interest". In this paper, based on the analysis of current component retrieval methods in web service and a method of retrieving software components based on domain ontology and user interest was studied and

implemented. This paper emphasized the definition of ontology feature domain model, the presentation of component description model based on ontology feature and the retrieval method of user interest. Based on these, they presented the component retrieval framework and an algorithm for retrieving related components. Finally, a component retrieval system was given, and an instance with components in E-Commerce field proved validity comparing with the retrieval methods based on keywords and facet.

Aboud *et al.* [47] premised a work on, "Automated architectural component classification using concept lattices". This paper discussed that, as the use of components grown in software development, building effective component directories. During the life-cycle of component-based software, several tasks, such as construction from scratch or component substitution, would benefit from an efficient component classification and retrieval. In this paper, they analyzed how classification of components can be done using their technical description (i.e. functions and interfaces) in order to help automatic as well as manual composition and substitution. The approach was implemented in the CoCoLa prototype, which was dedicated to Fractal component directory management and validated through a case study.
Peng *et al.* [48] presented a model on, "An Ontology-Driven Paradigm for Component Representation and Retrieval". In this paper, the key factors of component reuse were discussed and it was found out that component reuse is actually the reuse of knowledge about component. Component ontology was used to define the knowledge about component. Domain-specific terms were used to represent component by importing domain ontology into component ontology. In this paper component retrieving algorithm was implemented by ontology query and reasoning. This model was used in a large scale distributed simulation system and the fact revealed that component ontology was flexible enough for component reuse and efficiency of retrieving algorithm. Khode and Bhatia [49] presented a paper entitled, "Improving Retrieval Effectiveness using Ant Colony Optimization". They proposed a technique that helps-user to find the appropriate component and retrieve that. In their first step it matched keywords, their synonyms and their interrelationships. And then made the use of ant colony optimization, a probabilistic approach to generate rule for matching the component against the re-user query. The method showed very good values of precision and recall.

Aiming at the limitation of retrieving and extracting the most satisfying components in the component library, Meng Mei *et al.* [55] presented paper entitled "Research about Component Retrieval Based on Data Mining". In the paper firstly they discussed the application of classification method decision-tree-based for the component reuse. Secondly, they proposed the idea of applying data mining technology for the management of software component, which provided auxiliary decision support to the relevant personnel of the component library. D'Ambros and Robbes [56] presented a work on "Effective Mining of Software Repositories". Mining Software Repositories (MSR) is active and interest-growing research field-deals with retrieving and analyzing the data. They suggested that empirical analyses of software repositories allowed researchers to validate assumptions that were previously based only on intuitions, as well as finding novel theories. In turn, these theories about the software development phenomenon have been translated into concrete approaches and tools that support software developers and managers in their daily tasks. In this paper, they provided an overview of the state of the art of MSR. They described a different MSR approaches what techniques were available to researchers and practitioners, and finally, what the limitations of MSR were on that days, and how to fix them.

Software engineering principles are getting changed in order to serve various software development organizations. So the components developed for a software product may be useful for them if they develop similar product in future. Even though the components of a developed product related to one firm, else teams may require those components. So the components that are identified as re-usable are stored in a repository so that other teams can use them to serve in to get quality product. But to get the re-usable components from a repository there is need to search them effectively for getting needed component easily. The paper entitled "Searching Technique in Retrieving Software Reusable Components from a Repository" presented by Babu *et al.* [57]. They introduced a simple searching technique that may effectively retrieved required component from a repository. They used a web-based search to retrieve the desired component. For effective retrieval of component firstly there is need to search that needed component. Suffix Tree can be used for the same purpose. This data structure has been used in many applications like in effective retrieval of document, bioinformatics applications etc. It can also be used for document clustering. Chim and Deng [58] presented a paper entitled, "A New

Suffix Tree Similarity Measure for Document Clustering". In the paper, the authors proposed a new similarity measure to compute the pair wise similarity of text-based documents based on suffix tree document model.

4. Proposed Approach

The proposed architecture for the efficient search is given as in figure 4.1.Here, firstly user will enter the file name which act as repository in which he wants to search required component. After selection of repository, components that are presented in that repository are retrieved and their classification* is done on the basis of type of components, as in proposed work the concept of object oriented programming is taken, so classification is like whether component represents a class, a method or a attribute. Next step is their representation, as to retrieve the best component from the repository the foremost step is their effective representation. Here suffix tree is used for their representation, because it is one of the most important data structure which makes searching more refined and efficient. The next step is applying keyword search, user enter the component he wants to search and searching is takes place from suffix tree. At last the result of searching is provided to user i.e. whether the required component exists in repository or not. This process will be repeated for all files over the project.

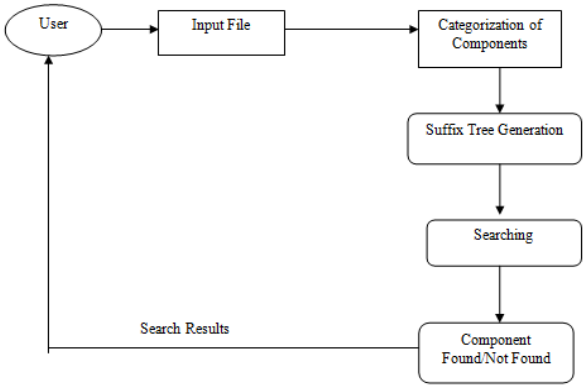

Figure 5.1 Flow chart for proposed work

4.1 Proposed Algorithms

Algorithm 1: Classification of Components

```
1.    Extraction Algorithm
2.    {
3.    Input the Program Filename called
      FName
4.    While s = GetLine(FName) <> null
      [Repeat Steps 3 to 12]
3     if Contains(s, "class")
4     {
5     Module = Module +1
6     Mname = Extract Module(s)
7     M = ToModule(Mname)
8     Methods = GetMethods(M)
9     TotalMethods = TotalMethods +
      length(Methods)
10    Attributes = GetAttributes(M)
11    TotalAttributes = TotalAttributes +
      length(Attributes)
12    }
13    Return (Module, TotalMethods,
      TotalAttributes,)
14    }
```

Functioning of an Algorithm: The above algorithm is for extraction of components from the required file which is act as repository here. Firstly user input the required filename called as FName and repeat the required process until the end of file. Algorithm read the file line by line and stored data contained in it in variable called 's'. Then it compare the value stored in 's' and if it is a 'class' then value of variable called module is incremented by one. Then it extracts all module names from s and stored it in another variable M. Then it make the use of standard library functions to extract all components from required file and increment the total method and total attribute by length of added method, attribute in it. At last algorithm return all modules, total attributes, total methods presented in the file which is input by the user.

Algorithm 2: Construction of Suffix Tree: This algorithm construct suffix tree of all above retrieved components.

```
Algorithm (ComponentName[],ComponentType[],N)
/*Here N is total number of components,
ComponentName contains list of all components and
ComponentType contains Type of components*/
1.    {
2.    Add Dummy Root Node to Tree, that
      Contains Node Information
3.    for i=1 to length(ComponentName)
4.    {
5.    comp = ComponentName(i);
6.    type = ComponentType(i);
```

```
7.      if (Tree=null)
8.      {
9.      if (type = "Module")
10.     {
11.     AddChild(Tree,comp,level);
12.     }
13.     }
14.     else
15.     {
16.     nodes = GetNodes(tree,level);
17.     for i=1 to length(nodes)
18.     {
19.     if(type(nodes(i))=type)
20.     {
21.     commonsuffix =
        FindCommonSuffix(node(i),comp)
22.     SetCenter(commonSuffix)
23.     SetRight(comp-commonSuffix)
24.     }
25.     }
26.     }
27.     }
28.     }
```

Functioning of an Algorithm: The above algorithm is for constructing suffix tree from all retrieved components. Here initialization is taken as null tree by adding dummy root node to it. From all the retrieved components from Algorithm1, it will extract component's name and type. If tree is null and type is 'module' then it make child from that module by adding level to tree, by default given algorithm initialized tree's level from 1 here. And if type is other then module, it simply adds nodes making further level of tree. At last common suffix is found which make center of the suffix tree and all different components is added to the right of suffix tree.

Algorithm 3: Searching Algorithm from Suffix Tree

```
SearchAlgorithm(Tree, Component)
        {
1.      While (Tree <> null)
        [Repeat Steps 2 to 23]
2       Node1 = GetNode(Tree)
3       Set Status = 0
4       If Type(Component) = Type(Node1)
5       {
6       M = length(Node1)
7       N = length(Component)
8       While (N>0 and M>0)
9       {
10      If (ExtractSuffix(Node1,M) <>
        ExtractSuffix(Component,N))
11      {
12      Status = 1
13      }
14      }
15      If Status = 0
16      {
17      Print "Component Not Found"
18      }
19      Else
20      {
21      Print "Component Found"
22      }
23      }
24      }
```

Functioning of an Algorithm : This Algorithm do searching of component entered from the user using suffix tree. Firstly it check whether tree is null or not, if it's not a null tree it goes on till the matching of all characters taking place from suffix tree. It initialize the status to 0, and then it match the type of 'component' (which is entered from the user for searching) with the type of node1(which is found from the tree by the help of GetNode function). It stores the total no of characters present in Node1 and in 'component' in two variables as M and N. While loop is executed till value of M and N is greater than 0. After retrieving total number of characters (in M and N), actual character matching takes place between the typed 'component' and obtained component as Node 1. If all the characters matched, it set status as 1 i.e. 'component found' else single mismatch of characters suddenly stop the working of this searching algorithm and print message 'component not found'. This is because suffix tree works on individual character matching, it stop matching further if single mismatch of characters takes place from suffix tree.

Conclusion and Future Work

The proposed work have aim to search the required component from repository with the help of new technique i.e. suffix tree. Suffix tree is one of the most vital data structures in string matching algorithms, so that searching would be efficient, hence retrieval of component. It is work on mechanism that during the matching of string if it doesn't found the characters in order, it stops the matching there and shows the results which makes the concept of searching more efficient and faster. In this present work, the retrieval of information regarding the software component from the existing software product is presented. This information is in terms of modules, methods and the attributes. The proposed system also provided the information regarding the availability of any existing software

component in the software product by using suffix tree. The implementation of proof-of-concept proved successful that the searching mechanism based upon suffix tree is capable of returning accurate search results. Suffix trees can be an alternative to various searching mechanism, but the advantages are fairly constrained. Here firstly all the components in the directory are retrieved, after that all the retrieved components are represented using suffix tree, possible candidate components are selected using keyword search.

The proposed searching technique based upon suffix tree is efficient and effective but still some future work is needed in this direction:

1. The work can be extended to generate the design view by using these components. The design view can be presented in the form of class diagrams, object diagrams and the process diagram.
2. In the present work an established technique to search a component from the repository provided the searching results effectively and speedily. But we are not concentrated on how to store reusable components in the repository. This can be done based on number of times a component is searched so far, so that we can have schemas in the repository for each and every component based on their frequent visit value.

REFERENCES

[1] Bourque,P., and Dupuis,R., "Guide to the Software Engineering Body of Knowledge", IEEE Computer Society, pp.1, ISBN 0-7695-2330-7, 2004.

[2] Arbab, B., and Berry, D.M., "Some Comments on 'A De-notational Semantics for Prolog'," 163-208, December, 1992.

[3] Bauer, Fritz., "Software Engineering: A Report on a Conference Sponsored by NATO Science Committee", NATO,1968.

[4] "Software Engineering ", A Volume of the Computing Curricula Series, The Joint Task Force on Computing Curricula IEEE Computing Society and Association for Computing Machinery, 2004.

[5] Ian Sommerville., "Software Engineering" 7th edition, Addison Wesley, pp. 1-31, 2004.

[6] R. S.Pressman., "Software Engineering-A Practitioner's approach" 6th edition, Mc Graw Hill, pp. 721-742, 2005.

[7] Debayan Bose., "Component Based Development", Application in Software Engineering Indian Statistical Institute, 2010.

[8] Mili,H.,Mili,A.,Yacoub, S., and Addy, E., "Reuse Based Software Engineering", Wiley-Interscience Publication, USA, 2001.

[9] Freeman,P., "Reusable Software Engineering: Concepts and Research Directions", 1983.

[10] Christine,B., and Marciniak, John J., "Encyclopaedia of Software Engineering", 1994.

[11] Krueger,C., "Software Reuse", ACM Computing Surveys, 1992.

[12] Peterson,A., "Coming to Terms With Software Reuse Terminology: A Model-Based Approach", 1991.

[13] Jacobson,I.,Griss,M., and Jonsson,P., "Software Reuse: Architecture, Process and Organization for Business Success", 1997.

[14] Brown, A.W., and Wallnau, K.C., "Engineering of Component Based Systems," Component-Based Software Engineering, IEEE Computer Society Press, pp. 7-15, 1996.

[15] Councill,B., and Heineman,G., "Definition of a Software Component and its Elements" , Component-Based Software Engineering: putting the pieces together, Addison-Wesley, Boston, 2001.

[16] Jacobsen,I.,Christerson,M.,Jonsson,P., and Overgaard,G., "Object-Oriented Software Engineering" Addison-Wesley, 1992.

[17] Booch,G., "Software Components With Ada", 1987.

[18] Meyer,B., "Rules For Component Builders" Technical report, 1999.

[19] Group,M., Meta group homepage, 1997.

[20] Szyperski, C., "Component Software: Beyond Object-Oriented Programming" ACM Press and Addison-Wesley, New York, 1998.

[21] Crnkovic,I., Larsson,S., and Chaudron ,M., "Component-based Development Process and Component Lifecycle" ,Software Engineering Advances ,International Conference , ISBN: 0-7695-2703-5,p-44, 2006.

[22] Mili, H., Mili, A., Yacoub, S., and Addy, E., "Reuse Based Software Engineering", Wiley-Interscience Publication, USA, 2002.

[23] Kaur,V., and Goel,S., "Facets of Software Component Repository", International Journal on Computer Science and Engineering(IJCSE), ISSN: 0975-3397,Vol.3,No.6, 2011.

[24] Henniger,S., "Supporting the Construction and Evolution of Component Repositories" , Proceedings of the 18th International

Conference on Software Engineering (ICSE), 1996.

[25] Naik,R., and Rao,M., "Information Search and Retrieval System in Libraries" 8th International CALIBER, 2011.

[26] Zhang,Z., "Enhancing Component Reuse Using Search Techniques" Proceedings of IRIS 23. Laboratorium for Interaction Technology, 2000.

[27] wikipedia.org/wiki/Suffix tree

[28] Zaremski,A., and Wing,J., "Specification Matching of Software Components" In Proceeding 3rd Symposium on the Foundations of Software Engineering (FSE3), p.17, ACM, 1995.

[29] http://www.cs.nthu.edu.tw/~wkhon/algo09/tutorials/tutorial-suffix-tree.pdf

[30] Gusfield,D., "Algorithms on Strings, Trees, and Sequences" Computer Science and Computational Biology. Cambridge University Press, 1997.

[31] Weiner,P., "Linear Pattern Matching Algorithms" In SWAT '73: Proceedings of the 14th Annual Symposium on Switching and Automata Theory, IEEE Computer Society,pp. 1-11, 1973.

[32] McCreight,E., "A Space-Economical Suffix Tree Construction Algorithm" ACM,Vol.23, pp. 262-272, 1976.

[33] Ukkonen,E., "Constructing Suffix Trees On-Line in Linear Time" In Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture Information Processing ,Vol.1,pp. 484-492, 1992.

[34] www.cs.nthu.edu.tw/~wkhon/algo09/tutorials/tutorial-suffix-tree.pdf

[35] http://homepage.usask.ca/~ctl271/857/suffix_tree.shtml

[36] Ning,J., "Component-Based Software Engineering (CBSE)", Assessment of Software Tools and Technologies Proceedings 5th International Symposium, ISBN: 8186-7940-9,pp. 34-43, 1997.

[37] Standish,T., and Thomas,A., "An Essay on Software Reuse", IEEE Transactions on software engineering, ISSN: 0098-5589, Vol. SE-10, No. 5, pp. 494-497, 1984.

[38] Chang,C., Chu,W.C., Liu,C., and Yang,H., "A Formal Approach to Software Components Classification and Retrieval", Proceedings 21st Annual International Computer Software and Applications Conference COMPSAC, ISBN: 0-8186-8105-5 ,pp. 264-269, 1997.

[39] Luqi., and Guo,J., "Toward Automated Retrieval for a Software Component Repository", Proceedings of IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), pp. 99-105, 1999.

[40] Lucredio,D., Gavioli,A., Prado,A.F., and Biajiz,M., "Component Retrieval Using Metrix Indexing" ,Information Reuse and Integration (IRI), Proceedings of IEEE International Conference ,pp. 79-84, 2004.

[41] Garcia,C., Lucredio,D., and Durao,F., "From Specification to Experimentation: A Software Component Search Engine Architecture", © Springer-Verlag, Berlin Heidelberg,pp.82-97, 2006.

[42] C. Veras,R., and Silvio,L., " Comparative Study of Clustering Techniques for the Organization of Software Repositories", ISSN : 1082-3409,Vol. 1,pp. 210 -214, 2007.

[43] Dixit,A., and Saxena,P.C., " Software Component Retrieval Using Genetic Algorithms" International Conference on Computer and Automation Engineering © IEEE, ISBN: 978-0-7695-3569-2, pp. 151-155, 2009.

[44] Ichii,M., Hayase,Y., Yokomori,R., Yamamoto,T., and Inoue, K., "Software Component Recommendation Using Collaborative Filtering", Search Driven Development Users, Infrastructure, tools and Evaluation SUITE, ISBN: 978-1-4244-3740-5, pp.17-20, 2009.

[45] Viana, T.B., Nobrega, H.I., Ribeiro, T., and Silveira, G., "A Search Service for Software Components Based on a Semi-Structured Data Representation Model", 6th International Conference on Information Technology: New Generations © IEEE, ISBN: 978-1-4244-3770-2, pp. 1479 -1484, 2009.

[46] Li,N,; and Luo,Z., "Component Retrieval Based on Domain Ontology and User Interest", EBISS International Conference © IEEE, ISBN: 978-1-4244-2909-7, pp.1-4, 2009.

[47] Aboud,N.A., Arevalo,G., Falleri,J-R., Huchard,M., Tibermacine,C., Urtado,C., and Vauttier,S., "Automated Architectural Component Classification using Concept Lattices", Software Architecture & European Conference on Software

Architecture WICSA/ECSA @2009 IEEE, ISBN: 978-1-4244-4984-2, pp. 21-30, 2009.

[48] Peng,Y., Peng,C., Huang,J., and Huang,K., " An Ontology-Driven Paradigm for Component Representation and Retrieval", 9th International Conference on Computer and Information Technology © IEEE, ISBN: 978-0-7695-3836-5, Vol. 2, pp. 187-192, 2009.

[49] Khode,S.G., and Bhatia,R., "Improving Retrieval Effectiveness using Ant Colony Optimization", International Conference on Advances in Computing, Control and Telecommunication Technologies © IEEE, ISBN: 978-1-4244-5321-4, pp. 737-741, 2009.

[50] Cai-lin,D., Zhen-zhen,Z., and Ying,Y., "Study On Component Hierarchical Retrieval Based On Ontology" 2nd International Conference on Future Computer and Communication(ICFCC) @IEEE, ISBN: 978-1-4244-5821-9,Vol.1, pp. 477-480, 2010.

[51] Niranjan,P., and Guru Rao,C., "A Mock- Up Tool for Software Component Reuse Repository", International Journal of Software Engineering and Applications (IJSEA),Vol.1,No .2, 2010.

[52] Shao,Y., Zhang,M., and Xu,S., "Research on Decision Tree in Component Retrieval", 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD) IEEE, ISBN: 978-1-4244-5931-5, Vol. 5, pp. 2290-2293, 2010.

[53] Kaur,V., and Goel,S., "Facets of Software Component Repository", International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol.3, No.6, 2011.

[54] Frakes,W., and Kang,K., "Software Reuse Research: Status and Future", IEEE Transactions On Software Engineering, ISSN: 0098-5589, Vol. 31, No. 7,pp. 529-536, 2005.

[55] Meng Mei., Baozhen Li., and Wei Ke., "Research about Component Retrieval Based on Data Mining", International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE), ISBN: 978-1-4244-9985-4, Vol.2, pp. 134 -137, 2011.

[56] D'Ambros,M., and Robbes,R., "Effective Mining of Software Repositories",27th IEEE International Conference on Software Maintenance (ICSM), ISBN: 978-1-4577-0663-9, p. 598, 2011.

[57] Babu,K., Kumari,K., and Rao,P., "Searching Technique in Retrieving Software Reusable Components from a Repository",International Journal of Scientific and Research Publications, ISSN 2250-3153,Vol. 2, Issue 2, 2012.

[58] Chim,H., and Deng,X., "A New Suffix Tree Similarity Measure for Document Clustering", Proceedings of the 16th international conference on World Wide Web, ACM, ISBN: 978-1-59593-654-7,pp. 121-130, 2007.