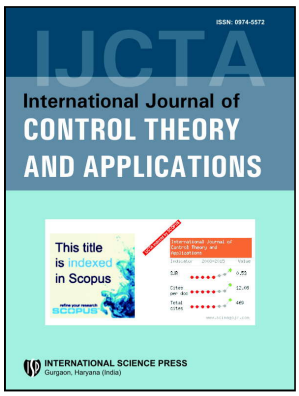


International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 19 • 2017

A Keyword Ontology for Retrieval of Software Components

Swathy vodithala[1] and Suresh Pabboju[2]

[1] *Assistant professor, Department of CSE Kits, Warangal, Telangana, India swathyvodithala@gmail.com*
[2] *Professor, Department of IT, CBIT, Hyderabad, Telangana, India plpsuresh@gmail.com*

Abstract: Software engineering is a wide domain which has a scope for emerging new technical research work with an integration of existing methods. In software engineering, Component based software development (CBSD) is one of the methodologies for developing the software. The main objective of CBSD is Software reuse, which uses the existing components rather than doing from scratch. The component can be a design, documentation or a piece of code and in the proposed work the component considered is a piece of code. This paper explains the proposed work in two phases, in first phase the description of component is done based on the keywords and the second phase explains the retrieval algorithm which retrieves the components from the repository based on the concept of ontology.

Keywords: CBSD, Software reuse, ontology, keyword, retrieval algorithm, semantics.

1. INTRODUCTION

The software engineering is an application of engineering to the software development. Component based software engineering(CBSE) or CBSD is a specialized methodology used to develop software in software engineering. Software reuse is a major objective of CBSE, which decreases the cost and time to develop the software. A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard. Software element contains a sequence of abstract program statements that describe the computation to be performed by a machine [1] [7].The component used in software reuse may be code, design pattern or even documentation. The reuse of the code i.e., code reuse is gaining major attention according to the survey when compared with design or documentation [8] [9][11].

The three major areas in software engineering which are to be focused while considering the components for software reuse are

a) Classifying or clustering the components.

b) Describing the components.

c) Finding or retrieving the appropriate components.

Finding the component from the software repository includes a variety of search techniques and retrieval techniques .It also includes querying [10] and indexing of the components in software repository.

The structure of the paper is organized as follows: The next section describes the related work, which has two techniques keyword search and ontology with its own advantages and disadvantages. The section III, explains the proposed algorithm based on ontology .In section IV, we have the experimentation results and later sections have conclusion and future scope.

2. RELATEDWORK

In order to attain efficiency in retrieving the software components, the software repository is to be well defined, structured and organized. Even though there are many retrieval techniques explained in the literature, only some of the techniques are explained in this section which are relevant to the proposed work.

2.1. keyword based search

The keyword search works with a principle of assigning some keywords which are relevant to the software component stored in the repository.Table-1 shows the database that stores the components and its corresponding keywords. The keywords are stored in the database by using delimiter #.

Example :

Table 1
Keywords of components

S. No	*Name of the component*	*Keywords*
1	Component 1	Kw1 # Kw2#KW3
2	Component 2	Kw1 # Kw2#KW4
3	Component 3	Kw3 # Kw4#KW5
4	Component 4	Kw6 # Kw7#KW8

If the user wants to search a component based on the keyword Kw3, then component1 and component3 are retrieved. The resultant may have many irrelevant components which makes keyword search to be on the weaker side [4][5].

2.2. Ontology

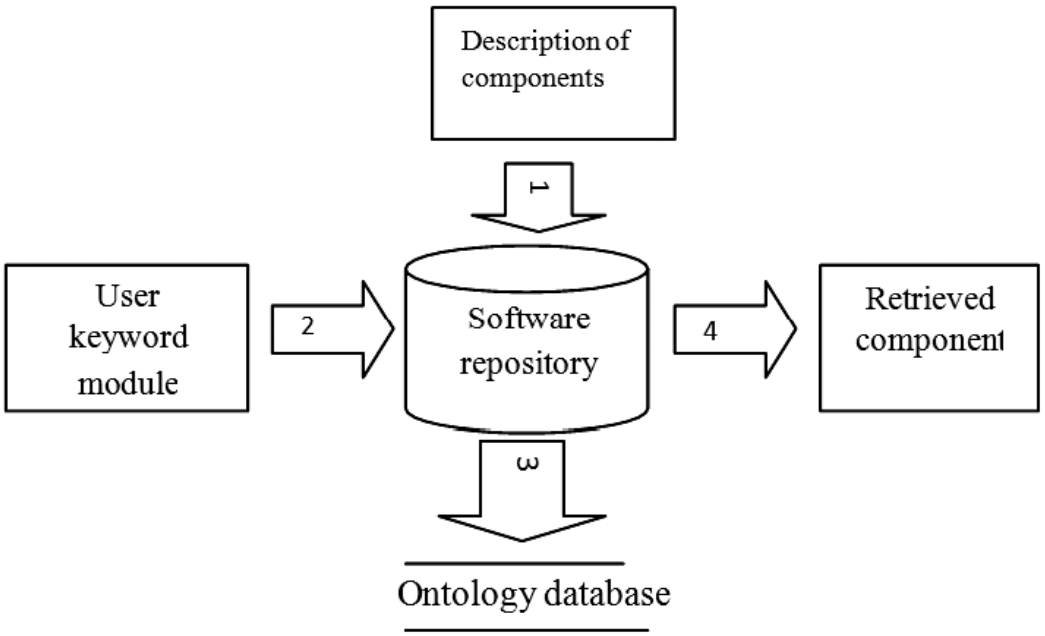
The ontology meaning in philosophy is considered to be as “theory of existence” [2]. Particularly, in computer science, ontology is formally describing the terms and the relationship among them in a particular domain. Ontology was initially used by AI laboratories but now it is gradually moving to the desktop of the domain experts [3].

There are many rules to build ontology. One such rule to build ontology is that, it should be a repetitive or iterative process. But the important thing to be considered while building ontology is that there is no exact way to model a particular domain, i.e. the best solution always depends on the application on which the user is working. The following ways can be used to build ontology for the Keywords [6].

1. Entity relation diagram
2. Listing
3. Tree representation
4. Predicate representation

3. PROPOSED WORK

The finding and retrieving of software components from the repository plays a vital role in CBSE. The proposed work is on code reuse therefore the software components which are stored in the repository must be code i.e., either functions or modules.



1. Storing into repository 2. User gives a keyword 3. Uses the ontology 4. Output

Figure 1: Proposed Architecture

Figure 1 describes the proposed architecture, where the repository consists of software components , the user can give a query and the extracted keywords from the query are stored. By using the ontology database and retrieval algorithm semantic keywords are returned by comparing the user keyword. Finally the relevant components are given as the output.

The proposed work can be described by two phases, where phase 1 shows the description of the component and phase 2 uses the concept of ontology. In the first phase the description of software component is done as shown in Table-2 and the procedure for describing the component is as follows: (i) define few keywords which are relevant to the component (ii) store the components along with the keywords in the database using a delimiter (iii) store the code in the database of a component so that it can be downloaded further.

In the second phase, an algorithm is described that is used for retrieving the software components based on the concept of ontology. The prerequisite for applying the proposed algorithm is that the ontology for all the keywords that are described in the table above must be described using tree representation.

Table 2
Component description

S. No	*Component name*	*Keywords*	*File stored in the repository*
1	Linear Search	Search#Linux#c	File1
2	Binary Search	Search#Sort#Cpp#Windows	File2
3	Bubble sort	Sort#windows#Java	File3

Fig 2 describes an example of creating the ontology for operating systems .In the tree representation if the user searches for the root node "windows" which is at level 0, then level 1 is considered as an exact match (i.e., windows Xp, windows 7, windows 8 and windows 10) and returns it as the result. If the user gives any one node from level 1 say windows 8, then all of its siblings are given as an exact match and further the level 2 is also returned as a relaxed match or approximate match for the node. If the user gives any one node from level 2 say Linux, then all of its siblings are given as an exact match and the level 1 is returned as the relaxed match. The same procedure is followed even if the number of levels in the ontology representation increases but it is important to note that as the levels in the tree increases the similarity from root node decreases.

Similarly, ontology is developed for all the keywords used to describe the component. Ontology for programming languages is designed and ontology for the keywords like search, sort is also semantically described. For an example consider another case where user enters a keyword "sort" which is a root node. The level 1 in ontology contains bubble sort, insertion sort, quick sort, heap sort and so on and these level 1 elements are returned as an exact match, further level 2 is returned as the relaxed match .The level 2 contains the keyword such as binary search because before binary search the process of sorting happens and thus can be considered as a relaxed match.

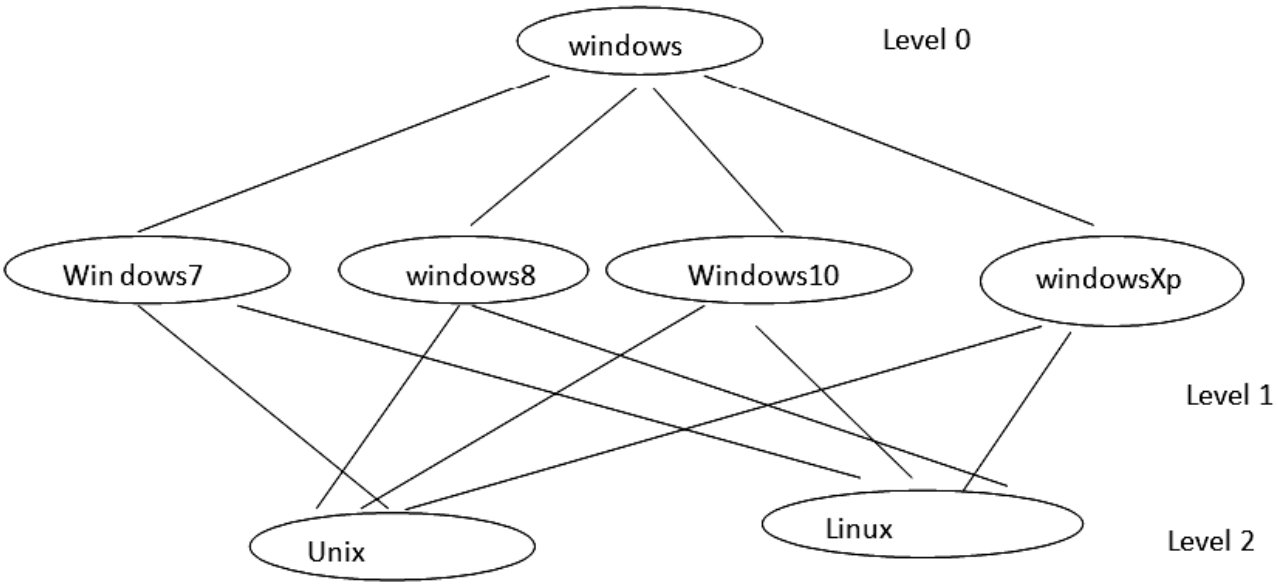


Figure 2: Ontology for operating systems

Algorithm for retrieving of software components using keyword ontology:

Input: All the components in the repository and user keyword

Output: Components relevant to user keyword

1. Begin
2. If the user keyword=root node ,then return elements of level 1 as exact match and level 2 as relaxed match.
3. If the user keyword=element from level 1,then return sibling elements as exact match and level 2 as relaxed match. Ignore the level 0 as it is root node.
4. If the user keyword=element from level i , then return sibling elements as exact match and level (i+1) and(i-1) as relaxed match, where i >= 2.
5. If the user keyword ≠ any element, return fail i.e., no keyword is matched.
6. End

After extracting the keywords which are semantically related by the user keyword, apply keyword search mechanism to retrieve the components from the repository.

The advantage of the proposed algorithm when compared with the keyword search is that the keyword search only retrieves the components which are assigned with the particular keyword, thus it does not give a scope of giving the appropriate or relaxed matches, whereas the proposed algorithm uses the concept of ontology in order to extract the keywords which are semantically related to the keyword given by user, thus can retrieve even the relaxed components rather than only retrieving the exact components.

4. EXPERIMENTATION RESULTS

Figure 3 shows an experiment conducted on a repository that consists of 300 components. Initially, keyword search was applied to the repository which resulted in 30% of the relevant components required by user. Later the keyword ontology was applied to the repository which yielded nearly 60% of the relevant components.

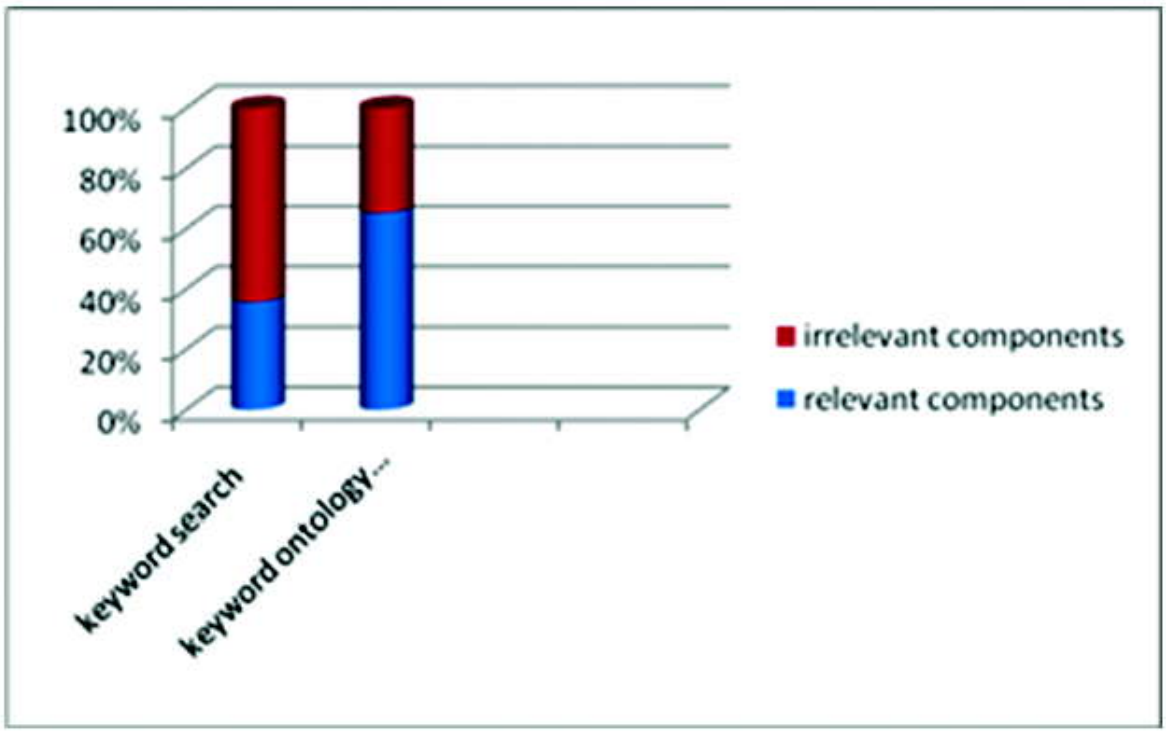


Figure 3: Comparision of keyword and keyword ontology search.

The relevance of the software components can be calculated by using the metrics precision and recall.

Recall = ({relevant documents} ∩ {retrieved documents})/{relevant documents}

Precision = ({relevant documents} ∩ {retrieved documents})/ {retrieved documents}

5. CONCLUSIONANDFUTURE SCOPE

There are many retrieval techniques explained in literature such as keyword search, faceted search, attribute based, signature and behavioral matching .All these mechanisms have their own advantages and disadvantages. Among the above mentioned techniques keyword search is very simple but requires an expertise to define the keywords of a particular component. The keyword search results only in the exact match i.e., the user given keyword are matched against the keywords stored for the component in the repository. This problem has been

avoided by the proposed retrieval technique by applying ontology to the keywords that can retrieve a set of keywords which are semantically related. Thus the proposed algorithm results in exact as well as relaxed matches for the components required by the user.

The future scope must be focused on decreasing the cost for building ontology because more cost is incurred in building the ontology. It would be even better to reuse the existing ontology required for the algorithm.

REFERENCES

[1] Swathy Vodithala, Niranjan Reddy, Preethi , "A Resolved Retrieval Technique for software Components" by IJARCET ,volume 1,issue 4, June 2012.

[2] Tom Gruber, "Ontology", entry in Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.) , Springer-Verlag, 2008.

[3] Natalya f. Noy and Deborah l. Mcguinness, "Ontology development 101: a guide to creating your first ontology", Stanford university, Stanford, CA, 94305.

[4] Rajender Nath and Harish Kumar, "Building Software Reuse Library with Efficient Keyword based Search mechanism", technia – International journal of computing science and Communication Technologies, Vol 2, No. 1, July 2009 (ISSN 0974-3375)

[5] Rajesh Batia, Mayank Dave and R.C.Joshi, "Retreival of most relevant reusable component using Genetic Algorithm", Scientific & Engineering Research, Volume 5, Issue 1, January-2014 , ISSN 2229-5518.

[6] Swathy Vodithala and Dr.Suresh pabboju, " A problem solver mechanism for mathematical text predicates", IJCTA 9(19), 2016, pp. 9297-9301, International Science Press.

[7] Book: "Definition of a Software Component and Its Elements" by Bill Councill, George T. Heineman

[8] Johannes sametinger, " Software Engineering with Reusable Components", Springer-Verlag, March 3, 1997.

[9] William B.Frakes and kyo kang, "Software Reuse Research:Status and Future", IEEE Transactions on software engineering, vol 31, No 7, July, 0098-5589.

[10] Yunwen Ye and Gerhard Fischer, "Supporting reuse by task-relevant and personalized information", International Conference on Software Engineering, Argentina, May 19-25, 2002.

[11] William W. Agresti, "Software Reuse: Developers Experiences and Perceptions", Journal of Software Engineering and Applications(JSEA), vol 4, No 1, Jan 2011.