Semantic-Based Retrieving Model of Reuse Software Component

Article · January 2010

CITATIONS
6

READS
153

3 authors, including:

Nedhal Al-Saiyd
Applied Science Private University
33 PUBLICATIONS **152** CITATIONS
SEE PROFILE

Intisar Al Sayed
Ashur University
10 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Semantic-Based Retrieving Model of Reuse Software Component

Dr. Nedhal A. Al Saiyd
Computer Science Department
Applied Science University
Faculty of Information Technology

Dr. Intisar A. Al Said
Computer Science Department
Al-Isra University
Faculty of Information Technology

Ahmed H. Al Takrori
Applied Science University
Faculty of Information Technology

Summary
As the demand of the software is increasing day by day, the software can be developed either from scratch or from using already developed software components. Component Based Software Engineering (CBSE) is known as a practical solution to the "Software Crisis". It improves productivity and quality of the developed software, but has extra time, effort and knowledge about identifying and extracting the reusable components from already developed and existing software systems
We investigate how ontology technologies can be utilized to support and identified relevant software component retrieving from open-access, different structured, very large and exponentially growing repositories on WWW. The system employs a natural language understanding for the user query to find the conceptual intention, and as the ontology allows word meaning to be queried, it is possible to formulate the unstructured natural language user query into well-defined conceptual query.
The component ontology consists of knowledge about the reuse component: functionality, structure, interfaces; requires and provides interfaces, platform and the application domain from which the component is extracted. The component ontology comprises 33 categories of terms. A search engine that applies concept matching technique; enables the user to search for one or a combination of these tags within a component conceptual specification, Ontologies provide controlled vocabulary for the retrieving of reuse components. Our semantic-based approach makes the component retrieval more efficient and precise. It overcomes limitation of natural language's imprecision and then reduces the complexity of formal methods. We use description logics, which underlie Semantic Web ontology languages, OWL, to develop ontology for the matching components depending on ontological components descriptions.

Key words
Component Retrieval, Ontology, Reuse Component, component-based software engineering, Concept Matching Technique

1. Introduction

The demand for new software applications is currently increasing at the exponential rate, as is the cost to develop them. Component based software engineering (CBSE) is one of software process models and it is accepted as a powerful solution to the development of software. Software professionals have recognized reuse as a powerful means of potentially overcoming the software crisis and it promises significant improvements in software productivity and quality, and decrease the product development life time and product cost. 65% of typical software is made up of domain-specific class of software. So the most savings is expected, if the domain specific software is reused. It means one should concentrate on evaluating the software in terms of its relevancy to a particular domain [1].
A Software component is a unit of composition with well-specified interfaces that show what these component present to the world and explicit context dependencies only. Software components are designed to be used as a plug and playable. But in reality they are not able to provide this functionality because not all components have compatibility with each other. Components need a platform on which they can stand and able to work together [2,3]. The following figure shows the architecture of software component.

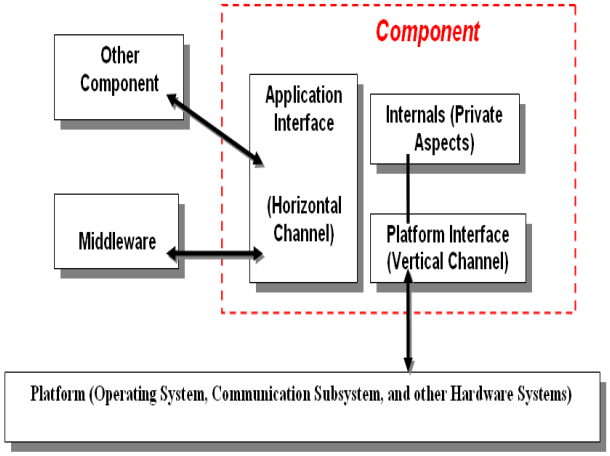

Fig. 1 The architecture of software component

2. Motivation and Problem Statement

Component-based software engineering (CBSE) increases the productivity, reliability and maintainability of software through reuse. There is huge number of reusable components that were individually developed, tested and stored in different-structured repositories on a World Wide Web. The major problems that are related to CBD is [4,5]:

1. The component-based developer needs to search for and retrieve a relative software component(s) that is well-matched with his required specification. The

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

components are stored on distributed, different structured component repositories on Web. The
2. Select the most appropriate component out of many founded.
3. Compose the most appropriate founded component with other developed components after adaptation (if needed) to construct new software application of specific domain.

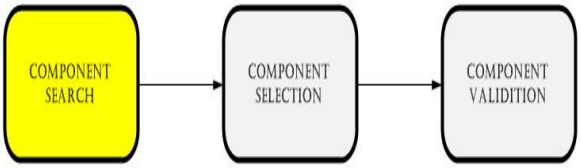

Fig 2 The Component Identification Process

The developer can retrieve a limited set of components and do not consider semantic relationships between components. Reasoning about component descriptions and component matching is a critical activity in Web-based component development.
Another problem is that not all features are important for software applications. Some of the features may be redundant or irrelevant. Some may even misguide the searching result, especially when there are more irrelevant features than relevant ones. In such case, selecting a subset of original features often leads to better searching performance. Feature selection not only reduces the high dimensionality of the feature space, but also provides better data understanding, which will improve the searching result.

3. Solution Proposal

The solution is based on ontology-based component repository and using formal axioms that represent more information on concepts and their relationships as well as restrictions related to properties and concept values. Formal axioms capture richer semantics and can contribute towards retrieving semantically interrelated software components. The aim is to collect, represent the shared conceptualization (semantics) of both the component-based developer's queries and software components specifications in the ontological searching and retrieving of relevant software components that uses the semantic similarity technique.
Ontology languages and theories of the Semantic Web can be adopted. The Web ontology language (OWL) is equivalent to very expressive description logic. Description logics provide a range of class constructors to describe concepts.

4. Ontology-Based Representation

Many references define ontology as: a formal representation of the basic terms that comprise the vocabulary of a specific domain, relations that form association between terms, and the set of axioms; which are the rules and constraints for combining terms and relations to define extensions to the vocabulary. It is the model of the concepts that is used to reason about the properties of the knowledge domain. Hence, ontology is used by people, databases, and applications that need to share domain information [6,7,8].
For retrieving component-based reuse software components, we have to reveal the semantics for the text of component-based Developer's queries and components specification as they are published on the Web. Normally, there are several preprocessing steps for the representation model and retrieval methods.

4.1 Defining and Extracting Software Component Features

The selected feature set should contain sufficient and more reliable information about the component data set. This will be formulated into the problem of identifying the most informative words within a set of documents that are associated with software components. Feature selection can improve the efficiency and accuracy of searching and retrieving by removing redundant and irrelevant terms from the corpus.
Features are relatively easy to specify, but they don't fully capture the semantics. Other nonfunctional semantics are supported by letting the user sort the resultant software components by code size, code complexity, or performance.
The categories of the identified metadata that are important in the context of component development and deployment and that lending to adequate support for the retrieval process, are:

a. Application domain ontological metadata [9]; describe the Application family domain of the software under development and the domain of software application from which the reusable component is extracted.
b. *Software development ontologies* describe the software development entities and processes, and.
c. Component model ontologies that defines core properties of a software component [10]. It involves:
i. Syntactic Specification: specifies the syntax of using component's services (technologies as COM or JavaBeans), specifies the semantics of these services (CBSE methodologies), and specifies properties besides component's services.
- A component implements a set of named interfaces and can require a set of interfaces implemented by

others. These are called Application (Horizontal) Interfaces that represent requires and provides interface. They show the ordering of operation activations that a component user has to follow meaningfully and consistently, as shown in fig.3.

- Platform (vertical) Interface represents the operating system, hardware, communication system.
- An interface implements a set of named operations
- An operation has a set input and output parameters with associated types, and data format.

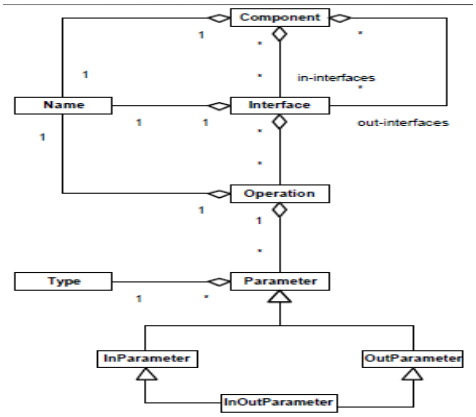

Fig 3 Syntactic Specification

ii. Component Semantic Metadata *Content Metadata*); is an extension of syntactic specification; where a state model is associated with each interface and operations have pre - and post-conditions (i.e. Abstract behavior) and Intra-interface conditions for components, as shown in fig 4. Semantic metadata contains information extracted from the source code or from component documentation [10].

iii. In addition, we extract other important features that are related with reuse software components as: quality factors; programming language; component size; code complexity; software design model, license, its developer; development year …etc.

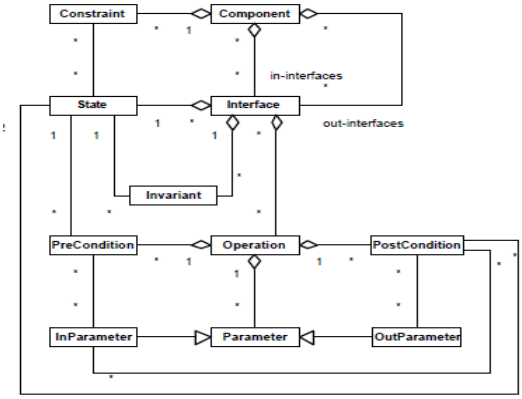

Fig 4 Semantic Specification

4.2 Conceptual Graph to Map Ontologies

Ontology-based terms analysis make use of internal structural. Ontology is a "specification of a conceptualization", whereby a conceptualization is a way of thinking about this domain. Ontologies belong to the knowledge representation approaches and they aim to provide a shared understanding of a domain both for the computers and for the humans [11]. Thereby, ontology describes a domain of interest in such a formal way that it can be processed by computers. A conceptualization; which has a hierarchical order, is a collection of objects, concepts, other entities and their associated relationships, that are recognized to exist in domain of reusable software components and their repositories. The conceptual graph (CG) model and can be used to formalize the information in the reuse component underlying domain. CG organizes and converts an informally perceived view of a domain into a semi-formal specification, using a set of graphical notations that can be understood by domain experts and ontology Developers. Fig.5 shows the conceptual graph of component specification.

Any wrong choice in this process will set off cascade of errors into the succeeding modules and probably prevent them from making the right choice.

4.3 Construction of Feature Vector (FV)

The following steps are proposed to find the FV of the different application domains using training software components specification:

1. Extraction of Meta Information: Meta information is collected from the sample of 50 software components in form of identifiers/keywords and identifier-by software matrix is created. The useless identifiers are removed and normalization is performed.

2. Perform Similarity Analysis: Similarity analysis between FV of the potential Reusable Component and the FV of component extracted from different application domains is performed and the similarity vector tells the relevancy level with the specified software component domains (i.e. comparison of items without reasoning) Fig.6, and transformed the compound concepts through the ontology into set of similar concepts.

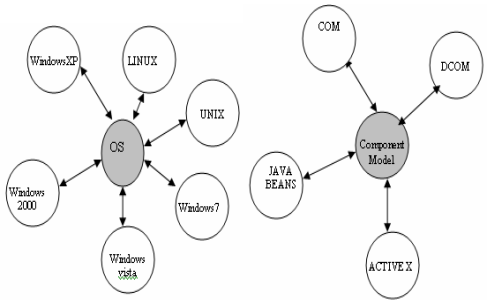

Fig.6 the synonyms words representation and the relation represents ' is-a'

It is possible for our system to locate terms which do not even appear in a component specification. Components which are located in a similar part of the concept space (i.e. which have a similar meaning) are retrieved, rather than only matching keywords, as shown in fig.7. By using a concept space, following problems can be solved.

a. *Polysemy*, or the problem that most words have more than one meaning, and that meaning is obtained from the word's context.
b. *Synonymy*, or the problem that there are many ways of describing the same object. The presence of synonyms tends to decrease the *Recall* performance of Information Retrieval systems.

Typically, a knowledge base consists of ontology, some data and also an inference mechanism. The inference mechanism would deploy rules in form of axioms, restrictions, logical consequences and other various methods based on the formal definition in the ontology over the actual data to produce more information out of the existing one.

5. Our System Architecture

The overall functionality of the system is divided into modules with a well-defined interfaces and activities. For this reason, the object-oriented design paradigm is used. The object-oriented design has encapsulation property that hides the inner details of the modules from other modules, and the common interfaces define how to use the modules and access their information. A module can depend on other module thus forming the architectural design. The modules are communicated using common interfaces. Therefore, the overall architecture of our system has *three* major phases, as it is shown in fig. 7; which are:

5.1 Query Evaluation

It processes the component-based developer's query asking about a required component. Our approach of retrieval can be seen as an evolution of classic keyword-based retrieval techniques, where the keyword-based index is replaced by a semantic knowledge base. There is no need to add weights automatically or manually by the user; to indicate the relative interest of the user for each of the tokens to be explicitly mentioned.
To reduce the gap between user intension and the system interpretation of queries and component specification, a set of similar concepts are used as ontological similarity technique [12]. The idea is to map the information found in the query and component specification into an ontology and be closer to the meaning of information. It is done as a part of the query evaluation in order to reduce the response time of retrieving the intended components. The interpretation of the given query is expanded with closely related concepts in order to achieve match with a conceptual description of an intended components rather than specific words or concepts [13]. This will compensate the ambiguous senses in natural language. Concepts are closely related when they have high degree of similarity and that are positioned closely together in the ontology with respect to distance. Similarity is subjective criteria and needed to be empirically. The query structure will be represented as a set of set (list of words) structure:

$Q= \{D_1,D_2, \ldots..,D_n\}$
$= \{\{d_{11},d_{12}, \ldots.,d_{1k1}\}, \ldots , \{d_{n1},d_{n2}, \ldots,d_{nkn}\}\}$

Where:
D_i's are a set of descriptors d_{ij}, j=1, ….k_i, and
$\{d_{i1},d_{i2}, \ldots.,d_{ik1}\} \in D_i$

$WordForm_1$	$WordForm_2$	$WordForm_3$	.....	$WordForm_n$	Word Meaning
$WF_{1,1}$	$WF_{1,2}$	$WF_{1,3}$	...	$WF_{1,n}$	M_1
$WF_{2,1}$	$WF_{2,2}$	$WF_{2,3}$	...	$WF_{2,n}$	M_2
$WF_{3,1}$	...	$WF_{3,3}$	...		....
...	..		...	...	...
$WF_{m,1}$	...	...	...	$WF_{m,n}$	M_m

Fig.8 Lexicalized concepts for the words and the associated meaning

Our system uses inference mechanism for implicit query expansion based on class hierarchies and rules (e.g. Java can satisfy a query for programming language). The query is executed against the knowledge base, which returns a list of features and their related semantic similarities that satisfy the query.

5.2 An Ontology-Based Knowledge Base

As the volume of the available information is increasing rapidly, it is in turn makes it difficult for human to browse through or manipulate them. Apparently, the reason is that information is currently represented in a semantically poor format, which means it is easily understood by people but not by machines. In contrast, semantic technology empowers the computer systems by enabling them to represent the information in semantically rich format, which means easily understood by computers [14]. Utilizing semantic technology, machines will be able to extract meaning from the information and to process them in an automatic fashion, with less human involvement.
The ontological knowledge in retrieving process has two different aspects. One deals with the ontology; as a goal of the query evaluation to retrieve knowledge instead of retrieve the components specifications and the other one involve the use of the ontological knowledge structure to reason, and to navigate the domain; which is covered by the ontological knowledge base for components specifications.

RDF knowledge base is constructed for components specification. Although XML is a suitable format for exchanging data, it only represents the syntax of the domain data but not the semantic of it. In contrast, the Web Ontology Language (OWL) is capable of adding semantic to data. Furthermore, using an appropriate semantic web framework, knowledgebase with OWL [15] format can be queried and reasoned over. These capabilities are convincing reasons to nominate OWL to propose Procedural Reasoning System PRS-style agent architecture [16] that have to deal with ontological knowledge base and semantic agent. Therefore, consistency, adaptability and generality are three main characteristics that using ontological approach to PRS. The entire domain knowledge is represented semantically using OWL ontology Fig. 9 shows a piece of ontological knowledge base coded with OWL/RDF languages [16].

5.3 Component Searching using Semantic Matching

The main purpose of introducing ontologies is to move from a query evaluation based on words to a query evaluation based on semantic, thus moving from syntax to semantics interpretation. The semantic matching [17] is used to find the strongest semantic relation between software component objects and queries; depending on a semantic basis; to retrieve the relevant components. It performs the actual comparison between description of query and description of software components. It computes the degree of similarity. It takes into consideration if a concept is: part-of another concept, consists-of another concept (generality), equivalent-of another concept, or disjoints-with (mismatch) other concept. All of these are done using semantic rules. The semantic relations are identified in the rule mining process that uses feature selection process. The feature selection process is applied on each ontological component specification. Finally, the software components that are annotated with these features are retrieved, and presented to the user.

6. Evaluation of Developed System

Recall and Precision have long been used to assess the quality of searches [18]. It is tried to evaluate the system in terms of these criteria. Let *S* be a set of all software components contained in a repository. *Precision* is the fraction of how well a retrieved set of components are relevant to the CB developer's need. Its formula is:

$$Precision = \frac{\sum_{s \in S} precision_{soft}(s)}{|S|} \quad (1)$$

Where:

$$precision_{soft}(s) = \frac{|C_{Actual}(s) \cap C_{Ideal}(s)|}{|C_{Actual}(s)|} \quad (2)$$

and *Recall* is defined as:

$$Recall = \frac{\sum_{s \in S} recall_{soft}(s)}{|S|} \quad (3)$$

Where:

$$recall_{soft}(s) = \frac{|C_{Actual}(s) \cap C_{Ideal}(s)|}{|C_{Ideal}(s)|} \quad (4)$$

Where $C_{Aactual}(s)$ is a set of software component containing software "s", generated by our search engine and $C_{Ideal}(s)$ is a set of technically relevant software component containing software "s", determined manually by the domain experts. Using *Precision* and *Recall* values F-value is calculated as a measure of performance evaluation

$$\text{F-Value} = \frac{2pr}{p+r} \quad (5)$$

Where *p* is the *Precision* and *r* is the *Recall* of the system.

7. Conclusion

In the traditional methods, the search engines for the user's query processing are based mainly on literal matching of keywords to retrieve software components specification. Their performance is limited because the conceptual meaning of the keywords and their synonyms are not applied. Therefore, an ontological-based model; which is based on semantic matching, is used to overcome these drawbacks, to improve their performance to retrieve the relevant software components. The experimental results demonstrate that ontology-based searches generate significantly better results than traditional search methods.

1. It is used to bridge the gap between software engineering (especially CBSE) and AI techniques through the use of ontologies of knowledge sharing in Knowledge-Based Applications KBA, and through the use of UML class diagrams in the development of ontologies.
2. The ontological knowledge base is designed and constructed using ontological concepts, structure and terminology of reuse component specification. Initially, we extract the concepts from 50 software components.
3. Our research helps the researchers and scientists to conceptualize the knowledge of reuse software component in component-based development, in order to develop a universal ontology, and try to avoid any inconsistencies and ambiguities that are commonly produce in defining and representing terms and concepts.

4. Reduce terminological and conceptual mismatches, by forcing to share understanding and communications among different users during the ontological analysis.
5. The general architecture of our system is generated using UML. UML is designed to build models by human experts. Then the transformation of UML to OWL/RDF file is generated. OWL is used at runtime by intelligent processing methods.
6. Improving the retrieval process using semantic matching and similarity measures. It helps avoiding the retrieval of irrelevant components.

8. Future work

1. Automatically extracting reusable software components information from the Web using *Text2Onto* tool to store the extracted information in the ontological knowledge base.
2. Measure performance of searching for a large number of software components information stored in ontological knowledge base, after adding new specifications Collect and add more than 50 components to cover almost all software components.
3. Measure and compare our system with other systems. Study the other related approaches and compare the results obtained with results of other component repository approaches.
4. Provide agent-based that uses ontological knowledge representation to solve problems in other areas.

Acknowledgements

The authors are grateful to the Applied Science University in Amman, Jordan for the partial financial support granted to cover the publication fee of this research article.

References

[1] Parvinder Singh Sandhu, Janpreet Singh and Hardeep Singh, Approaches for Categorization of Reusable Software Components, Journal of Computer Science 3 (5): 266-273, 2007
[2] Councill , and Heineman G. T., Component-based software engineering: putting the pieces together, Addison-Wesley Longman Publishing Co, 2001.
[3] www.ceng.metu.edu.tr/~oguztuzn/publications/02f-siw-co.pdf
[4] Heineman, G., Councill, W, Component-Based Software Engineering, Addison Wesley, 2001.
[5] Pressman R., Software Engineering A Practitional's Approach, Sixth Ed., McGraw Hill, 2005.
[6] Hans-Jörg Happel and Stefan Seedorf, Applications of Ontologies in Software Engineering Lindeberg, T., Feature detection with automatic scale selection. International Journal of Computer Vision, 30: 77-116, 2004.
[7] Dragan Ga˘sevi´, Nima Kaviani, and Milan Milanovi, Ontologies and Software Engineering, nima.magic.ubc.ca/www/2publications/2008/.../PDF_Version.pdf
[8] Collaborative Software engineering chapter 6: Application of Ontology in Collaborative Software Development, Springer-Verlag, 2010
[9] Su W., Wang J. and Lochovskyode F. H.: Ontology-Assisted Data Extraction,
[10] Klein K., Defining Software Component Specifications: An Open Approach, NDIA Systems Engineering Conference October 22-26, 2007, www.dtic.mil/ndia/2007systems/Wednesday/AM/.../5526_Kelin.pdf
[11] Cristiane A. Yaguinuma, Marilde T. P. Santos, and Marina T. P. Vieira, Ontology-Based Meta-model for Storage and Retrieval of Software Component,
[12] Liu, H. and L. Yu, Toward integrating feature selection algorithms for classification and clustering. IEEE Transaction on Knowledge and Data Engineering, 17: 491-502, 2005.
[13] F.A. Grootjen and Th.P. van der Weide, Conceptual query expansion, Data & Knowledge Engineering 56(2) (2006), 174–193.
[14] John F. Sowa, Conceptual Graphs For Representing Conceptual Structures, http://www.jfsowa.com/pubs/cg4cs.pdf.
[15] OWL Web Ontology Language Guide http://www.w3.org/TR/2004/REC-owl-guide-20040210/
[16] Hadzic, M., P. Wongthongtham, T. Dillon and E. Chang, 2009. Ontology-Based Multi-Agent Systems. Studies in Computational Intelligence. Vol. 219, Springer-Verlag Berlin Heidelberg, ISBN: 978-3- 642-01903-6, pp: 3.
[17] F. Giunchiglia, P.Shvaiko, M. Yatskevich: S-Match: an algorithm and an implementation of semantic matching. In Proceedings of ESWS'04.
[18] http://trec.nist.gov/pubs/trec10/appendices/measures.pdf

Dr. Nedhal Al Saiyd. She got her B.Sc. degree from University of Mosul-Iraq in 1981, M.Sc. and PhD degrees from University of Technology, Baghdad-Iraq in 1989 and 2000 respectively. She is an Assistant Prof. at Computer Science Dept., Faculty of Information Technology, in the Applied Science University, Amman, Jordan. Her research interests include Software Engineering, Ontology Engineering, Intelligent Systems, User Authentication, and Speech Processing.

Dr. Intisar Al Said. She got her B.Sc. from University of Technology, Baghdad-Iraq, in 1986, M.Sc. degree from Al-Nahreen University, Baghdad-Iraq, in 1993, and PhD from University of Technology, Baghdad-Iraq in 2000. She is an Associated Prof. at Computer Science Dept., Faculty of Information Technology, in Al-Isra University, Amman Jordan.

Her research interests include Genetic Algorithms, Intelligent Systems, Neural Networks, Ontology Engineering, Computer and Control Systems.

Ahmed Al-Takrori is a student in Department of Software Engineering at Applied Science University, Amman, Jordan. His research interests are: Intelligent Systems, Software Engineering, and Ontology Engineering.

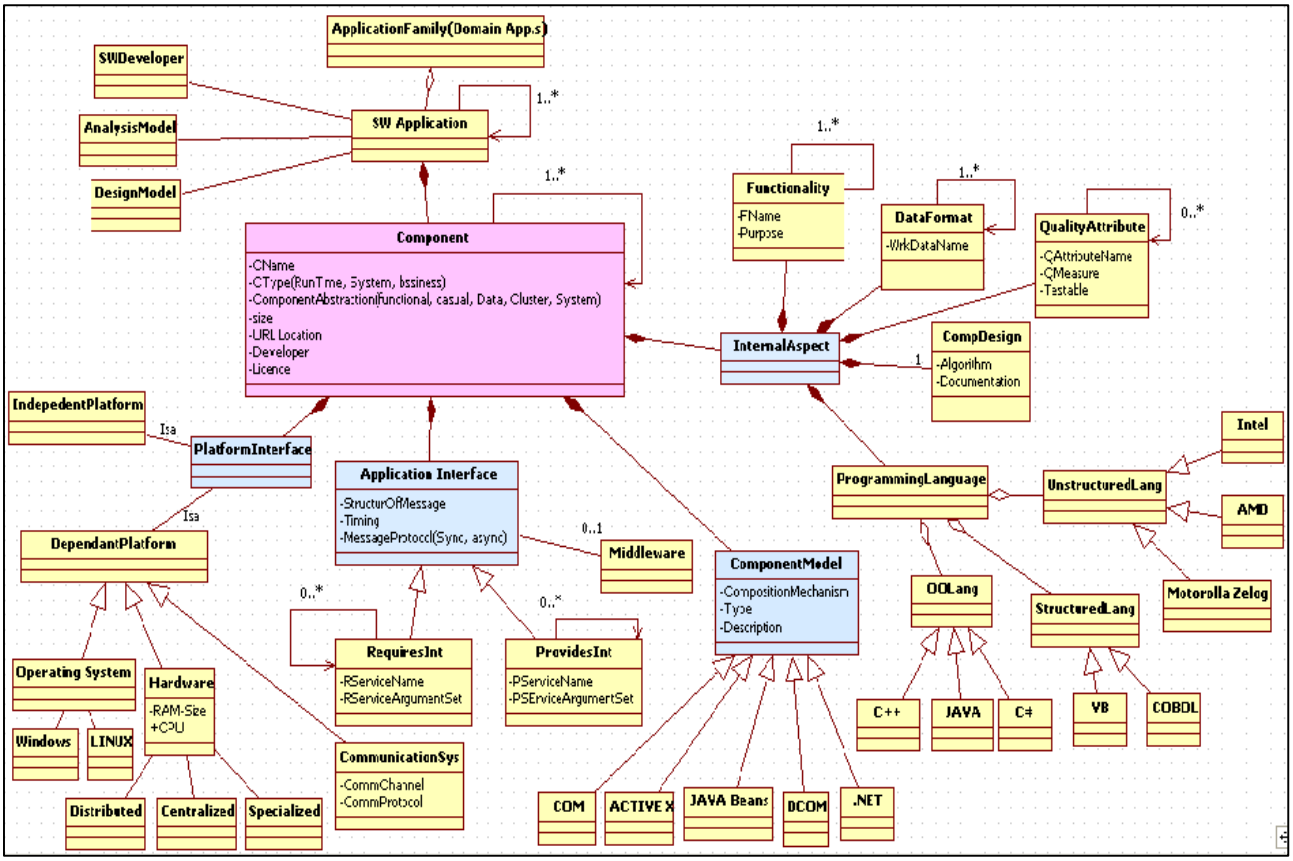

Fig 5 The Conceptual Graph for component specification

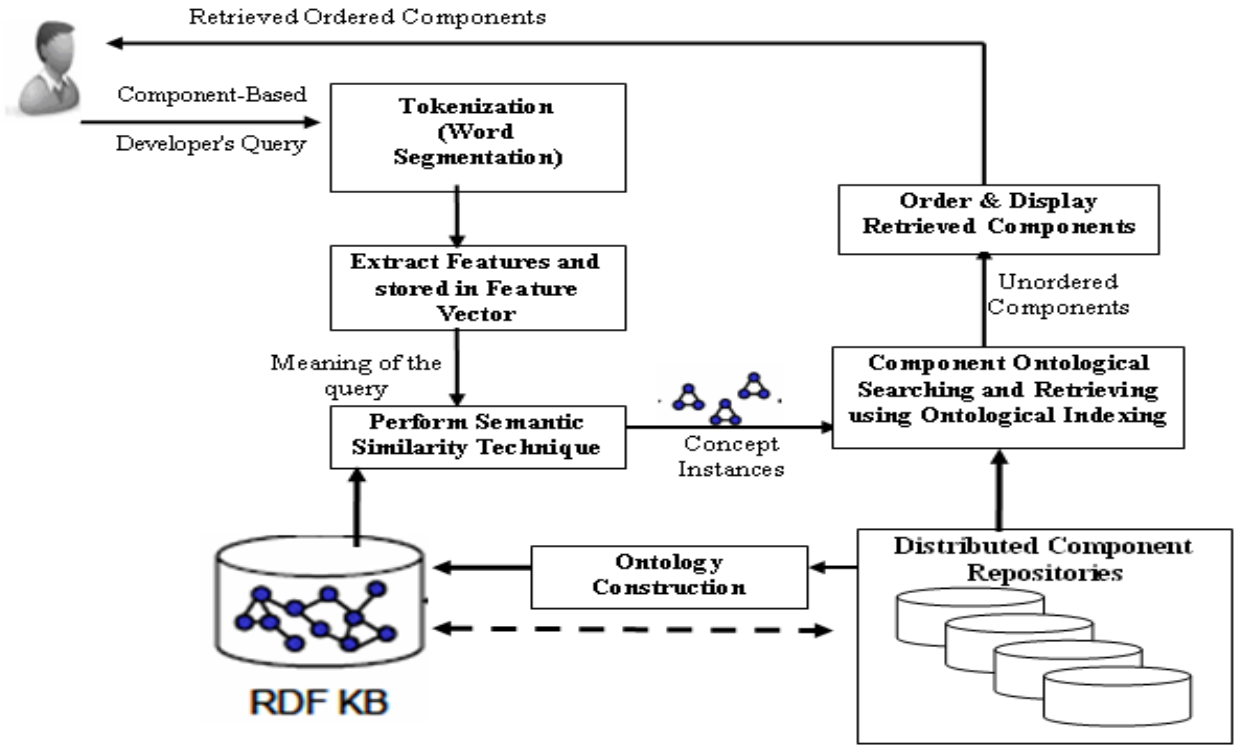

Fig.7 Ontology-based Component Retrieving

```
<owl:Class rdf:ID="analysis_and_design_approach">
      <rdfs:subClassOf rdf:resource="#speclized_domain"/>
   </owl:Class>
   <owl:Class rdf:ID="analysis_and_design_methodology">
      <rdfs:subClassOf rdf:resource="#speclized_domain"/>
   </owl:Class>
   <developer rdf:ID="apache_common"/>
   <owl:Class rdf:ID="application_family"/>
   <owl:Class rdf:ID="architecture_model">
      <rdfs:subClassOf rdf:resource="#speclized_domain"/>
   </owl:Class>
   <programming_language rdf:ID="asp.net">
      <used_for_developing rdf:resource="#x360_multiple_video_player"/>
   </programming_language>
   <speclized_domain rdf:ID="audio"/>
   <owl:InverseFunctionalProperty rdf:ID="avilable_in">
      <rdf:type rdf:resource="&owl;ObjectProperty"/>
      <rdfs:domain rdf:resource="#quality_factors"/>
      <rdfs:range rdf:resource="#component"/>
   </owl:InverseFunctionalProperty><component rdf:ID="avis_map"> <has_comment
rdf:datatype="&xsd;string"uct extends the Viewer by adding an extensive set of features for editing,
digitizing, merging, exporting, converting (between formats), building/correcting
map topology, etc. GIS shapefile map data.</has_comment>
```

Fig. 9 a piece of the implemented ontological knowledge base coded with OWL/RDF languages