Test Case Reuse Based on Ontology*

Lizhi Cai[1, 2], Weiqin Tong[1], Zhenyu Liu[2], Juan Zhang[1]

[1]School of Computer Engineering and Science, Shanghai University, Shanghai, China
clz@ssc.stn.sh.cn

[2] Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai, China

Abstract

Test cases are one of the most important assets in the testing process. This paper presents the testing ontology based SWEBOK and software quality model. The management and retrieval of test cases will play a vital role in test cases reuse. The keyword-based, as well as facet-based retrieval cannot meet user's flexible query requirement because of lack of semantic information. SWEBOK provides a broad agreement on the content of the software engineering discipline. At last this paper discusses the management and retrieval of test cases based on the semantic similarity of two test concepts in two ontologies according to difference sets of super concept, sub concept, extension, and intension.

Keywords: test cases, retrieval, ontology, reuse, similarity

1. Introduction

Software defects almost exist in any softwares with moderate size because the complexity of software is generally intractable and humans have only limited ability to manage complexity. A study conducted by NIST in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually [1]. More than a third of this cost could be avoided if better software testing was performed. Discovering the defects in software is equally difficult for the same reason of complexity. Software testing is difficult, time-consuming, and expensive. Typically, more than 40% percent of the development time in software lifecycle is spent in testing.

In software engineering, software reuse has been thought as a key strategy for reducing development costs and improving quality [2]. Recent years, software reuse has been promoted by object-oriented method and component-based development method. However, the reuse research of software test case, the most important assets in software testing process, is just beginning. A test case that finds a new error has proven to be a valuable investment [3]. The excellent test cases will be stored in the library as a reusable resource after being designed for reuse in the future. Among the test cases reuse process, a key question is the effective test cases organizing and management to satisfy the requirement of reuse, especially the description and retrieval of test cases. The current retrieval of resource is based on attributes-value, catalog structure, keyword and so on. These retrieval methods are the lack of semantic information. The system cannot carry out expansion or reasoning based on semantic to meet user's query request flexibly. Description and retrieval based on facet is mostly widely used in software component management [4,5]. Facet-based as well as traditional keyword-based methods, retrieval operations are carried out based on the vocabulary. There is a certain degree of semantic missing. Although the term dictionary can provide semantic information, but in most cases can only describe some simple relationships such as synonyms.

Ontology as a basis for the sharing of knowledge has been widely used in information science, such as software reuse [6], information retrieval [7], and requirement elicitation[8]. This paper, supported by the reusable test cases library project from the National High Technology Research and Development Program of China, carried out the test cases reuse research based on ontology.

2. Ontology

The word ontology has been taken from Philosophy, where it means a systematic explanation of Existence. It had been adopted by early Artificial Intelligence researchers, who recognized the applicability of the work from mathematical logic. In1993, Gruber defined an ontology as follows [9] "an ontology is an explicit specification of a conceptualization". This definition is the most referenced in the literature. Ontology is formalized using five kinds of components: classes, relations, functions, axioms and instances. It can be described in a 5-tuple.

$$< Class, Function, Relations, Axiom, Instance >$$

The relationships widely used include: part-of, kind-of, instance-of, and attribute-of. In the actual ontology model, the relationships between different

This work is supported by a grant from the National High Technology Research and Development Program of China (863 Program) （No. 2007AA010401 ）

concepts are not limited to four kinds of basic relations listed above. Specific relation should be defined in accordance with the specific circumstances to meet the needs of applications. For example, in software testing area, Test Engineer will use performance tool. Test engineers A and B are colleagues. The term "***use***" and "***coworker-of***" can be defined to describe the relation in testing action as follows:

Test-Engineer ***use*** performance-tool
EngineerA ***coworker-of*** EngineerB

Relations can also exist between classes, individuals, class and subclass, class and individual. An ontology sample about engineer and performance tool is shown in figure 1.

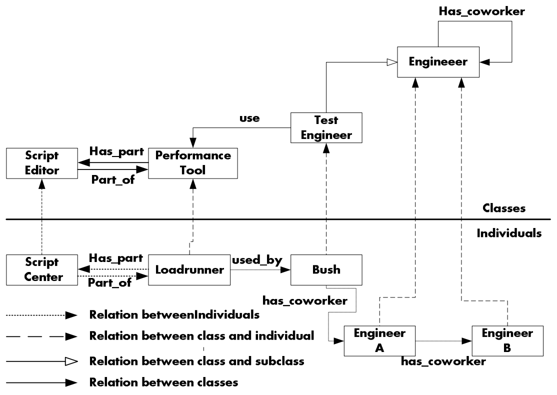

Fig.1 Relations between test engineer and performance tool

Ontology could play an important role in software testing area, as it do in other disciplines, where it: 1) provides a source of precisely defined terms that can be communicated across people such as tester, developer, QA, manager and user. 2) offers a consensual shared understanding concerning the software testing discourse; 3) renders explicitly all hidden assumptions concerning the objects pertaining to software testing knowledge.

In order to avoid conversion between different description languages, ontology needs for a common standard language to express. Because XML has been the standards on data exchange, some description languages based on XML such as OWL (Web Ontology Language) and RDF (Resource Description Framework) has been developed, especially in semantic web stack[10]. The OWL is a family of knowledge representation languages for authoring ontology, and is endorsed by the World Wide Web Consortium. OWL ontology is most commonly serialized using RDF/XML syntax.

The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph. It can be illustrated by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link (hence the term "graph"). The RDF graph of ***use*** relation mentioned earlier is shown on figure 2.

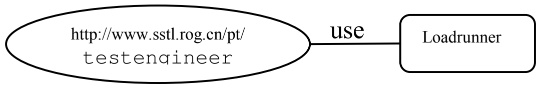

Fig. 2 RDF graph of ***use*** relation

The triple above can also be equivalently represented in the standard RDF/XML format as:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22
-rdf-syntax-ns#"
   xmlns:pt="http://www.sstl.org.cn/pt#">
<rdf:Description
    rdf:about="http://www.sstl.org.cn/pt/t
estengineer">
   <pt:use>Loadrunner</pt:use>
```

3. Software testing ontology

The Guide to the Software Engineering Body of Knowledge (SWEBOK) is a significant milestone in reaching a broad agreement on the content of the software engineering discipline[11]. A SWEBOK user is not only interested in the definitions of the concepts, but also in much more detailed information about the topics that are important to him. Researchers has developed some ontology from different viewpoint, such as quality ontology[12] , Software artifacts and activities ontology[13], software engineering teaching[14].

To provide a topical access to the knowledge, each knowledge area is further broken down into topics and subtopics, and identifies as well the related seminal reference material and a matrix linking the reference material to the topics listed. Testing ontology provides a vocabulary for representing and communicating knowledge about software testing and a set of relationships which hold among the concepts in software testing. The concept of testing, together with its set of multiple subconcepts, is the fourth knowledge area in SWEBOK. It includes software testing fundamentals, test levels, test techniques, test related measures and test process five sub-areas. The sub-area software testing fundamentals presents the testing-related terminology, key issues of testing, and the relations between testing and other activities. Test levels subarea is divided between the targets and the objectives of the test. Test techniques subarea is divided into subconcepts, according to different basis, such as the tester's intuition and experience, specification-based, code-based, fault-based, usage-based and the application nature based. Test related measures are grouped into those related to the evaluation of the program under test and the evaluation of the tests performed. Test process subarea includes practical considerations and the test activities.

We use “skeletal” methodology to build the testing ontology based on SWEBOK. A skeletal methodology for building ontology has been proposed and tested by Uschold and King [15]. The process of constructing software testing ontology has two ways: top-down and bottom-up. In the upper SWEBOK used to build the basic ontology of software testing. In the actual work process, the concept cluster will be formed by synthesizing the software testing concepts, terms. The concept cluster will be mapped into the upper layers of software concept. The final testing ontology constructed is shown on figure 3.

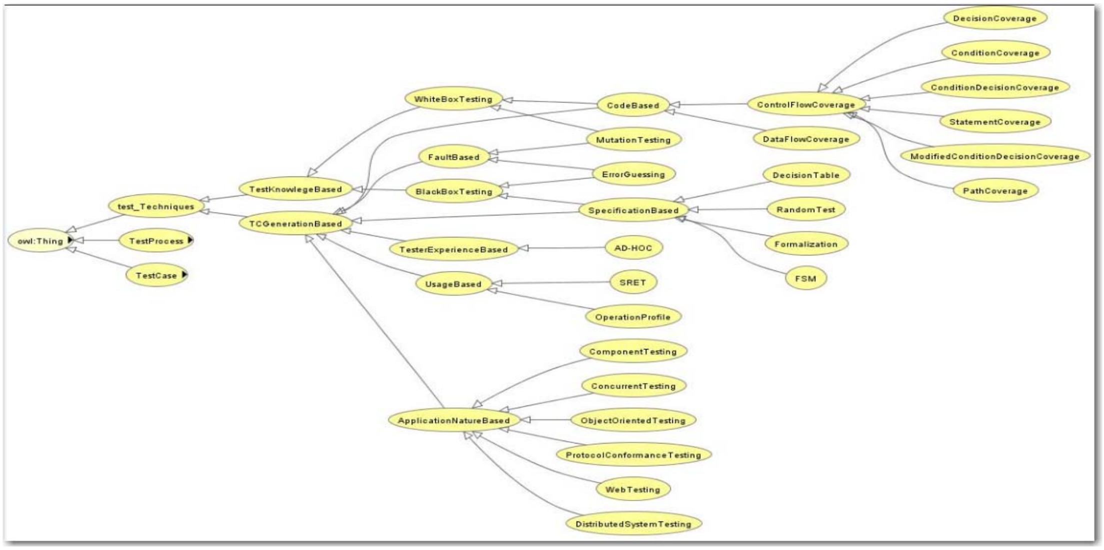

Fig 3 Software testing ontology

The subdomain partition of test cases can further enhance reusability of use cases, as well as provide a convenient sharing in a large-scale. A test case classification specification is developed by the collaborative effort among all enterprises who are taking part in building the reusable test case library. The specification is based on categories of software and coding delivered by National Bureau of Statistics of China and software classifying and coding guidelines in computer software copyright registration. It is organized in two different levels. The first level is based on software functionality and usage. The second level is based on ISO 9126 software quality characteristics[16]. ISO 9126 is an international standard for the evaluation of software quality. The fundamental objective of this standard is to address some of the well known human biases that can adversely affect the delivery and perception of a software development project. ISO 9126 tries to develop a common understanding of the project's objectives and goals. The software testing classification ontology is shown as figure 4.

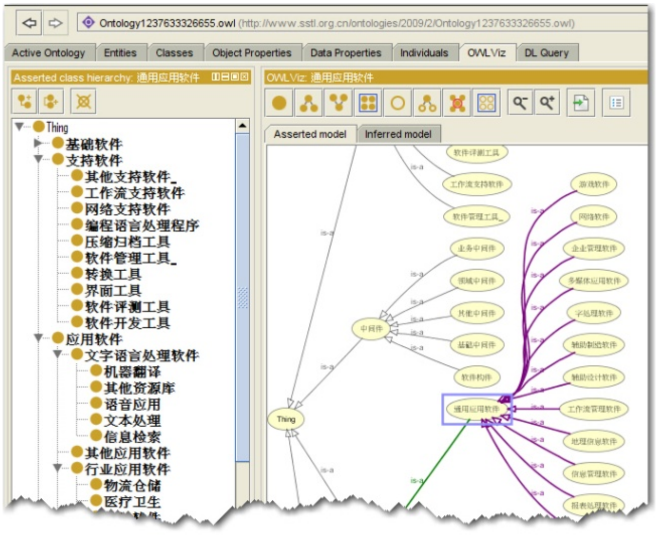

Fig. 4 Software testing classification ontology

4. Test case reuse

The similarity and relatedness of ontology play a vital role in test case search and matching process. The terms car and gas tank are more closely related in the relevance, but car and bicycle are more similar. Automatic performance testing and automatic function testing have concept similarity but less relatedness, loadrunner and performance testing have more semantics relatedness. Semantic distance can be used to describe semantic similarity. We will use the concept of the semantic distance to calculate semantic similarity indirectly. The semantic distance of testing concept can be computed by four difference sets of two corresponding concepts in different ontologies[17]. They are super-concept difference set D_{Sup}, sub-concept difference set D_{Sub}, , intension difference set D_{Int} and extension difference set D_{Ent}. The semantic distance $Dist(c, c')$ can be calculated from the following formula.

$$Dist(c, c') = \beta_1 D_{Sup} + \beta_2 D_{Sub} + \beta_3 D_{Int} + \beta_4 D_{Ext}$$

Here $\beta_1, \beta_2, \beta_3, \beta_4$ is the weight of different factors.

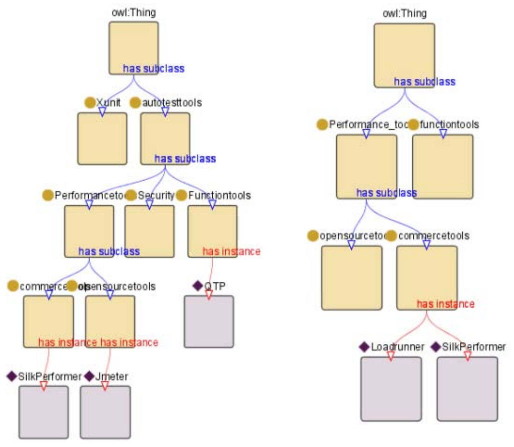

Fig. 5 Two performance tool ontology

An ontology *O* about automatic testing tools are shown in left side of figure 5. *Performancetools*, which is identified as *c*, is a concept of ontology and *C* is the concept set of ontology *O*. The concept *Performancetools* has two data properties. The first property is *license* which indicates the number of concurrent virtual user. The second property is *supportweb* which indicates whether the tools support the web application performance testing. The value of *license* is 250 with data type xsd:int. The value *supportweb* is true with data type xsd:boolean . The core code of property of concept *Performancetools* is defined as follows.

```
<owl:DatatypeProperty rdf:ID="license">
  <rdfs:domain rdf:resource="#Performancetools"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf>
        <rdf:List>
          <rdf:first rdf:datatype="&xsd;int">
              500
          </rdf:first>
          <rdf:rest rdf:resource="&rdf;nil"/>
        </rdf:List>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="supportweb">
 <rdfs:domain rdf:resource="#Performancetools"/>
 <rdfs:range>
   <owl:DataRange>
       <owl:oneOf>
       <rdf:List>
          <rdf:first rdf:datatype="&xsd;boolean">
               False
          </rdf:first>
          <rdf:rest rdf:resource="&rdf;nil"/>
       </rdf:List>
     </owl:oneOf>
   </owl:DataRange>
 </rdfs:range>
</owl:DatatypeProperty>
```

In the other ontology O' , concept *Performancetools* also has two properties *license* and *supporttuxedo*. The property *license* has the same meaning as ontology O with value 500. The property *supporttuxedo* indicates the whether the tools support tuxedo protocol with value true. The RDF/XML code of ontology O' is similar to the ontology O. The super concept set $Sup(c)$ consists of all the concept's ancestors. The sub concept set $Sub(c)$ consists of all the concept's descendants. The extension of a concept $Ext(c)$ consists of all the instances. The data property of a concept *Con(c)* consists of all data property. These sets can be defined as:

$$Sup(c) = \{a \mid a \in C, c\ \boldsymbol{kindof}\ a\}$$

$$Sub(c) = \{a \mid a \in C, a\ \boldsymbol{kindof}\ c\}$$

$$Ext(c) = \{a \mid a \in C, a\ \boldsymbol{instanceof}\ c\}$$

$$Con(c) = \{a \mid a \in C, a\ \boldsymbol{datapropertyof}\ c\}$$

In these formula, ***kindof*** expresses relationship between the concept and sub concepts or between the concept and super concepts. For two concept *a* and *c*, the formula *c* ***kindof*** *a* indicates *c* is sub concept of *a*, including direct and indirect blood relationship. In the ontology O, the concept *Performancetools* is a sub concept of *autotesttools*. The relation ***intanceof*** indicates the relation between concept and it's instance. The relation a $\boldsymbol{dataproperty of}$ c indicates a is one of the data property of c. *SilkPerformer* is an instance of *commercetools.* The concept *Performancetools* in ontology O and O'respectively is denoted by c and c'. We can get the value of $Sup(c)$, $Sup(c')$, $Sub(c)$, $Sub(c')$, $Ext(c)$, $Ext(c')$ as follows:

$$Sup(c) = \{autotesttools; Thing\}$$
$$Sup(c') = \{Thing\}$$
$$Sub(c) = \{opensource; commercetools\}$$
$$Sub(c') = \{opensource; commercetools\}$$
$$Ext(c) = \{Jmeter;\ SilkPerformer\}$$
$$Ext(c') = \{Loadrunner;\ SilkPerformer\}$$
$$Con(c) = \{licence, supportweb\}$$
$$Con(c') = \{licence, supporttuxedo\}$$

The super concept of *Performancetools* in ontology O consists of autotesttools and OWL:Thing. The super concept of *Performancetools* in ontology O' is OWL:Thing. Both of them have same sub concepts *opensourcetools* and *commercetools*. The intension of concept *Performancetools* in ontology O has two instances *Jmeter* and *SilkPerformer,* which are respective instance of *opensourcetool* and *commercetools* . *Performancetools* in ontology O' has two instances *loadrunner* and *SilkPerformer* which are instance of *commercetools*.

We can define the difference set of super concept $D_{Sup(c,c')}$, the difference set of sub concept $D_{Sub(c,c')}$, the difference set of extension $D_{Ext(c,c')}$, and $D_{con(c,c')}$by unified formula $D_{Set(c,c')}$.

$$D_{Set(c,c')} = 1\text{-}\frac{|Set(c) \cap Set(c')|}{|Set(c) \cup Set(c')|}$$

$$= \begin{cases} D_{Sup(c,c')}\ here\ Set(c) = Sup(c), Set(c') = Sup(c') \\ D_{Sub(c,c')}\ here\ Set(c) = Sub(c), Set(c') = Sub(c') \\ D_{Ext(c,c')}\ here\ Set(c) = Ext(c), Set(c') = Ext(c') \\ D_{Con(c,c')}\ here\ Set(c) = Con(c), Set(c') = Con(c') \end{cases}$$

The values of $D_{Sup(c,c')}$, $D_{Sub(c,c')}$, $D_{Ext(c,c')}$ can be calculate by these formula:

$$D_{Sup(c,c')}=1\text{-}\frac{|\{Thing\}|}{|\{autotesttools, Thing\}|} = \frac{1}{2}$$

$$D_{Sub(c,c')}=1\text{-}\frac{|\{opensource, commercetools\}|}{|\{opensource, commercetools\}|} = 1$$

$$D_{Ext(c,c')}=1\text{-}\frac{|\{SilkPerformer\}|}{|\{Jmeter, Loadruner, SilkPerformer\}|} = \frac{2}{3}$$

$$D_{Con(c,c')}=1\text{-}\frac{|\{licence\}|}{|\{license, supportweb, supporttuxedo\}|} = \frac{2}{3}$$

Ontology has two type properties, data property and object property. Object property describes the relationship between objects, and data property describes the values of the properties. The similarity of intension can be calculated through the difference set of data property and the value difference of data property. The $D_{Int(c,c')}$ be defined as follows:

$$D_{Int(c,c')} = \alpha D_{Con(c,c')} + \beta \frac{\left(\sum_{s_i \in s_1 \cap s_2} d_i(v_i, v_i')\right)}{|s_1 \cup s_2|}$$

The parameters α and β is weight of two part. s_1 and s_2 are respectively property sets of c and c'. The value of $d_i(v_i, v_i')$ can be calculated according to the type of data property. When the data type of data property is boolean or enumerate, it can be calculated by:

$$d_i(v_i, v_i') = \begin{cases} 1, v_i \neq v_i' \\ 0, v_i = v_i' \end{cases}$$

If the data type of data property is integer, $d_i(v_i, v_i')$ can be calculated by:

$$d_i(v_i, v_i') = \frac{|v_i - v_i'|}{max\,(|v_i|, |v_i'|)}$$

We can get the difference of property value of *Performancetools* in ontology O and O' according to the definition of $d_i(v_i, v_i')$

$$d_1(v_1, v_1') = 1, \qquad d_2(v_2, v_2') = 1$$

$$d_3(v_3, v_3') = \frac{|250 - 500|}{max\,(250,500)} = 0.5$$

Here v_1 , v_2 , v_3 stand for the *supportweb*, *supporttexudo* and *license* property. Suppose $\alpha = 0.5, \beta = 0.5$, the intention difference of *Performancetools* can be calculate:

$$D_{Int(c,c')} = 0.5 \times \left(1 - \frac{1}{3}\right) + 0.5 \times \frac{1 + 1 + 0.5}{3} = \frac{3}{4}$$

Suppose $\beta_1 = 0.4, \beta_2 = 0.2, \beta_3 = 0.1, \beta_4 = 0.3$, we can calculate the value of *Dist*:

$$Dist(c, c') = 0.675$$

This can be drawn that the semantic similarity of *Performancetools* in ontology O and $O'$is 0.675. If the threshold of similarity for test cases retrieval is 0.5, the concept *Performancetools* in two ontologies are similar. When the value of $Dist(c, c')$ is larger than threshold, the concept being retrievaled will be accepted.

Ontology-based test cases reuse process includes some key steps such as testing ontology building, test cases collection, query pre-processing, query reasoning, ontology query generation and results return etc. The detail test case retrieval process shown in figure 6 is as follows:

(1) Build software testing ontology and testing classification ontology using Protégé with the help of domain expert.
(2) Collect software test cases and annotate them in the light of established ontology. The metadata produced by annotation is stored in the metadata base, and the test cases are stored in the test cases library.
(3) Preprocess the user's test case query request from user interface according to ontology. The result of reprocess will be sent to reasoner.
(4) Reasoner deduces over ontology by translating them to RDQL for test cases retrieval.
(5) Return the retrieval result to user through user interface.

5. Summary

As software becomes increasingly pervasive, the need for quality and reliability in software systems continues to increase. Software testing is a very important phase of software engineering, whose cost is accounting for 40% total costs. As one of the most import assets in testing process, the reuse of test cases is attracting the attention of researchers. Facet-based as well as traditional keyword-based methods, retrieval operations carried out based on the vocabulary are lack of semantic information. SWEBOK provide a wide agreement on the content of software engineering. This paper presents the construction method of software testing ontology based on SWEBOK, and the software testing classification ontology based on ISO 9126 software quality model. After that, the management and retrieval of test cases based on ontology are discussed, especially the retrieval match similar of two concepts in two different ontology.

6. References

[1] NIST, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, http://www.nist.gov /director/prog-ofc/report02-3.pdf.

[2] Frakes W. Systematic Software Reuse: A Paradigm Shift. *In Proceedings of Third International Conference on Software Reuse: Advances in Software Reuse*. Los Alamitos, Califonria:IEEE Computer Society Press,1994

[3] Glenford J. Myers, *The Art of Software Testing, 2nd ed.*,John Wiley and Sons, Hoboken, N.J., 2004

[4] Xiaoqin Xie,lie Tang,Juanzi Li,Kehong Wang. A Component Retrieval Method Based on Facet-Weight Self-learning. *Advanced Workshop on Content Computing(AWCC 2004)*; 20041115-17, ZhenJiang, CN.

[5] Yang Yong, Zhang Weishi, Zhang Xiuguo, Shi Jinyu. A weighted ranking algorithm for facet-based component retrieval system. *In: ACST'06: Proceedings of the 2nd IASTED international conference on Advances in computer science and technology*, Puerto Vallarta, Mexico, 2006,pp274-279

[6] Yan Wu, Harvey Siy, Mansour Zand, Victor Winter. Construction of Ontology-Based Software Repositories by Text Mining. *In ICCS '07: Proceedings of the 7th international conference on Computational Science*, Part III. Beijing, China 2007,pp790—797.

[7] David Vallet, Miriam Fernández, and Pablo Castells. An Ontology-Based Information Retrieval Model. *Lecture Notes in Computer Science : The Semantic Web: Research and Applications*. 2006,pp455-470

[8] Lee Yuquin , Zhao Wenyun. An Ontology-Based Approach for Domain Requirements Elicitation and Analysis. *IMSCCS '06: Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 2 (IMSCCS'06)*. 2006,pp364-371

[9] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[10] Berners-Lee, Tim; James Hendler and Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved on 2008-03-26.

[11] Alain Abran and James W. Moore, Executive Editors, *Guide to the Software Engineering Body of Knowledge*, IEEE,2004, URL: http://www.swebok.org

[12] Wille, C., Abran, A., Desharnais, J.M. and Dumke, R. (2003) 'The quality concepts and sub concepts in SWEBOK: An ontology Challenge in Investigations in Software Measurement', *Proceedings of the 13th International Workshop on Software Measurement*, Montreal, Canada, pp.113-130.

[13] Miguel-Ángel Sicilia, Juan J. Cuadrado, Daniel Rodríguez: Ontologies of Software Artifacts and Activities: Resource Annotation and Application to Learning Technologies. *SEKE 2005*: pp145-150.

[14] Heidi J. C. Ellis, Gregory W. Hislop: An ontology for software engineering teaching modules. *International Journal of Metadata, Semantics and Ontologies,* 2(1): 2007,pp11-22

[15] Uschold, M. and King, M. Towards a Methodology for Building Ontologies.. *Presented at the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI95*, AIAI-TR-183, University of Edinburgh, Edinburgh EH1 1HN, 1995.

[16] ISO,*ISO/IEC 9126-1, Software engineering - product quality - Part 1: Quality Model, first ed*. 2001-06-15.

[17] Zhang De-hai, ZHU Yao, measuring semantic distance between concepts in ontologies, *computer science(In Chinese)* vol35 no. 9, 2008, pp 144-148.