Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2014, *6*, 237-243

Retrieving and Composition Method of Internetware Components

Jingxiao Lu[1,2], Chengming Zou[1], Jingyang Zhu[1,*], Luo Zhong[1] and Xi Zhang[1]

[1] *School of Computer Science and Technology, Wuhan University of Technology, Wuhan, 430070, China*

[2] *National Engineering Laboratory for Fiber Optic Sensing Technology, Wuhan University of Technology, Wuhan, China*

Abstract: With the rapid development of software reuse and globalization of the Internet, traditional software presents a new software paradigm which is called internetware. The emergence of internetware brings about new technologies to improve software reusability, cut costs in software development and increased development productivity. It is difficult to search the required internetware components and support the automatic internetware components composition in software system development. This paper presents an approach for classifying internetware of faceted classification, retrieving the required internetware components of tree matching, and then proposes a method that is based on concept of ontology to describe interfaces, used figure backward iteration to finish the composition. Simulation results show that the users can rapidly access the required internetware components and give the feedback of compositions to the users. It is proved that the retrieving method in this paper has higher recall and precision ratio.

Keywords: Composition, component retrieving, figure backward iteration, internetware, tree matching.

1. INTRODUCTION

With the development of the Internet, the application of computer software platform is developing from the traditional closed-end platform to open platform. The network platform has strong distribution, high degree of autonomy nodes and the connection environment is diverse and complex. In order to adapt to the complicated network platform, the software generates a new paradigm that is called internetware [1-4]. The internetware is not only able to perceive the changes of the external network environment, can also adjust itself in a certain degree according to the changes, as much as possible to meet the needs of users. The internetware has the characteristics of autonomy, cooperation, reactivity, evolution and polymorphism [4-6], which is different from traditional software.

It can effectively reduce the cost, improve the quality and the efficiency to develop a software system based on the internetwares. But in this process, there is a need to solve the following two questions: (1) how to find the components the users need in the internetware component library quickly; (2) how to combine the internetware components efficiently to achieve the needs of the users because single internetware cannot meet all the users' functions.

In this paper a new approach for retrieving the internetware components is proposed, classify the components by using the facet classification, describe them in XML language and search the required components through the method of tree matching. For the automatic composition, we present a method to describe interfaces of internetware components based on ontology in a uniform way for the interfaces matching, and combine the components by using figure backward iteration to get the required internetware components and give feedback to users.

2. OVERVIEW OF RELATED TECHNOLOGY

2.1. Faceted Classification

Faceted classification retrieval method [7] is an accurate method to classify the internetwares by the facets that reflect the essential characteristics of the internetwares. A faceted classification scheme is composed by a set of facets described the essential characteristics of the internetwares. Each facet classifies the internetwares in the virtualized storage pool from different perspectives and is composed by a set of terms. Descriptor is a set of terms from different facets, used to describe the internetware. Using faceted classification retrieval method can meet the users' needs of the internetware description, and be convenient to manage the internetwares.

2.2. XML

XML can adequately express the information in internetware because XML has a strong ability to express information and allows users to define tags. Using XML allows users to define tags, which best reflect the characteristics of the internetwares, to describe the internetwares. It can accurately define the types of data and the constraint relationships between information entities in the internetwares. It can contribute to the internetware information for rational organization and hierarchy. This allows the description of the information can be converted properly into the internetware facet description tree. This helps users to get the

1874-4443/14

needed internetwares when retrieving in the virtualized storage pool via the theory of tree matching. The following is an example of a brief description of an internetware faceted information via XML and the internetwawre tree of it as shown in Fig. (**1**).

<Internetware>
<Function>Measuring</Function>
<Run Environment>
<Operation System>Linux</Operation System>
<Database>Oracle</Database>
</Run Environment>
</Internetware>

2.3. Concept of Ontology

The concept of ontology [8, 9] from philosophy, is a theory of the existing problems. It has not been accepted with a uniform definition since it was introduced into the science field. The definition that was proposed by Gruber, "An ontology is an explicit specification of a conceptualization" is the most widely accepted. Ontology is used to describe the relationships between concepts in a certain area or even a wider range area. It makes the concepts and relations have recognition, clear and single definition within the shared range. Ontology has been widely applied in knowledge engineering, software reuse, information retrieval and heterogeneous information on the web processing, etc. Thus, using ontology to describe the internetware components can make a unified and standardized description of the interfaces in different components. It makes it easy for computer to combine internetware components automatically and provide the appropriate composition solution.

3. TREE MACHING

3.1. Concept of Tree

Tree is an N-limited set of data elements and a special case of figure. A tree can be described as the expression: $T = (V, E, root (T))$. In the expression, V is the node set of a finite number of data elements in the tree and root (T) belonging to V represents the root node of the tree. E represents the edge set in the tree that is a set of binary relation. It has the features of irreflexivity, antisymmetry and transitivity. If the relationship of $(u, v) \in E$ exists, say that node u is the parent of node v or node v is a child of node u, written as $u = parent(v)$ or $v = child(u)$. A tree can have only one root node. Each node has only one parent besides the root node. In addition, if the "siblings" in a tree are not ordered, this tree is a disorderly tree.

3.2. Tree Matching Model

Based on the tree match model, give a definition of tree match model [10] that adapts to the internetware component retrieving. Suppose that there are two trees, Q and T: Q is an internetware component query tree, and $Qsub$ is a sub-tree of Q; T is an internetware component tree, and $Tsub$ is a subtree of T.

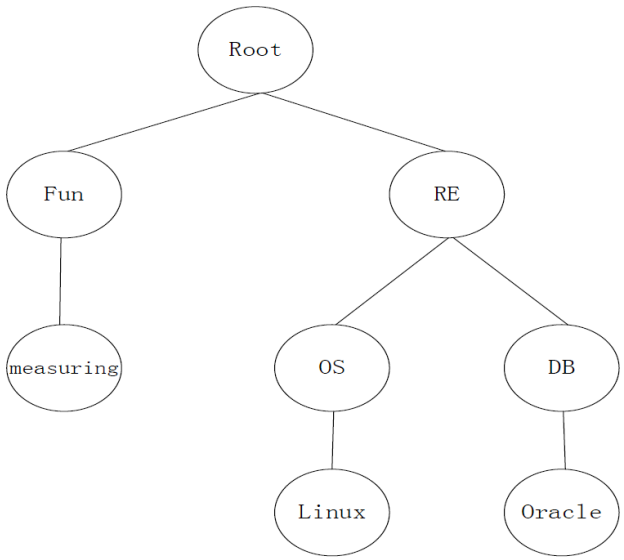

Fig. (1). Model of the internetware tree.

Definition 1 Path contained matching [11-13]: if there is a mapping named f from $Qsub$ to $Tsub$, f is considered to be a path that contains match when it satisfies the following three conditions.

(1) $v1 = v2 <=> f(v1) = f(v2)$; $v1$ and $v2$ are the nodes in $Qsub$, and this means that f is injective from Q to T;

(2) $v1 = anc(v2) <=> f(v1) = anc(f(v2))$; $v1$ and $v2$ are the nodes in $Qsub$, and this means that mapping f still keeps the relation of the ancestor and the offspring's between the nodes;

(3) $label(v1) \approx label(f(v1))$, this means the terms to describe the two labels are an approximation and within the acceptable range;

(4) The matched leaf nodes of Q exist in the T, and also are the leaf nodes in T.

The path contained matching of tree relaxes the correspondence relationship between nodes of the component trees. The only requirement is to ensure that the nodes between internetware component trees keep ancestor and offspring relationship and does not require a strict relationship of a parent and offspring between nodes. It allows some of the nodes involved in the node mapping to appear in different layers, improving the recall ratio in the query, shown in Fig. (**2**).

When retrieving the internetware components, the users care most about are the functions of components. In an internetware component tree, the leaf nodes represent the functions of the components. Therefore, in the use of the tree matching method to retrieve the components, matching the leaf nodes of the function sub-tree can finish the retrieving. So, an index is made of the leaf nodes of every component function sub-tree, and the component tree retrieval is done by this index.

Using a string represents the leaf nodes of the function sub-tree of each component tree, adding the root

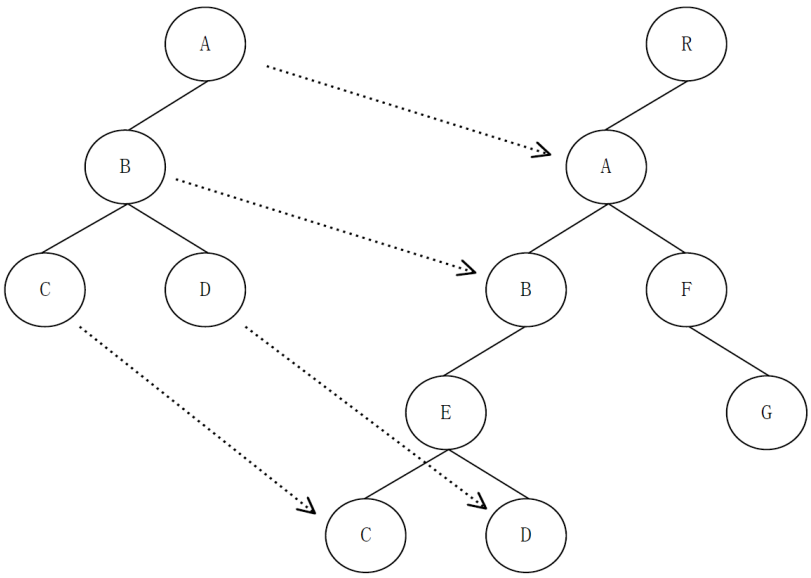

Fig. (2): The path contained matching of tree.

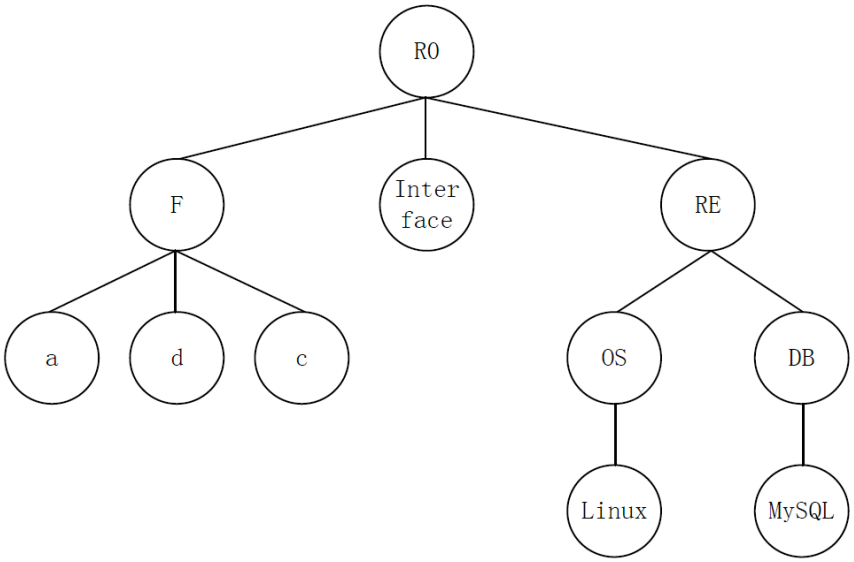

Fig. (3). The termed internetware component tree.

node of the tree to the end of the string retrieves the tree when the leaf nodes are matched. Therefore, the internetware components matching transforms to the problem of matching two strings.

3.3. Terminology Dictionary

Using terms to describe the facets of the internetwares is intuitive. The users often only care about the functions of the internetwares and ignore other features when retrieving the needed internetware components. So codes are made for the terms of the function leaf nodes in the component tree. During the term encoding, it is made certain that the code is the unique identifier of the term. The terminology dictionary is used to encode the terms. Terminology dictionary keep all the terms of the facets and the corresponding codes. Fig. (**3**) is a tree after encoding the function leaf nodes by the terminology dictionary and the function leaf nodes form a string of acd@R0.

3.4. Concept of Matching Degree

There may not exists an internetware that meets all the functions the user needs in the virtualized storage pool when retrieving the components. This causes a failure to match the strings. Therefore, in the matching process, a matching degree is used to calculate the similarity of the strings to find the component that have some functions the user needs.

Matching degree is used to calculate the degree when the characters in string *S1* match those in the string *S0*. Using matching degree can make similarity evaluation for the strings that are not an exact match. The matching degree is calculated as follows:

M= (S1.matchnum/S0.Length+S1.matchnum/S1.Length)/2

In the above formula *S1.matchnum* represents the number of characters in *S1* matching the characters in *S0*. The *S0.Length* and *S1.Length* represent the length of the string *S0* and *S1*. For example, the string *S0* is *acdfg*, string S1 is *abcfg*, and string *S2* is *acdglmnopq*. The matching degree of the *S1* and *S2* are 0.8 and 0.6, respectively. In this case, the matching character numbers of *S1* and *S2* for *S0* are equal. As the length of the *S1* is shorter, *S1* is more similar to *S0* than *S2*, so it has a high matching degree.

The higher matching degree of the string of the component tree means the internetware component has more functions for retrieval. After the introducing the concept of matching degree, use the components that have high matching degree to develop the system when no exact matching component exists.

4. MATCH OF INTERNETWARE COMPONENTS

The component retrieved by the tree matching cannot meet all the needed functions alone and the retrieved components to meet all the needs are combined. The composition of the internetware components is to make an order of the components execution and the internetware, consisted of these components, can finish the needed functions in the said order. But the description of the interfaces of the components may be different. This causes the semantic ambiguity in the interfaces of the components, leading to mistakes in the process of combining the components. So it is essential to make the semantics of the components' interfaces uniform to facilitate the combining of components.

Definition 2: For the Concept *A* and *B*, if *A* and *B* are defined as the equal semantics in the description of ontology, the relationship is written as $A = B$; if *B* include *A* in semantic, the relationship is written as $A\ B$. If the relationship of *A* and *B* accords with the above relationships, it is called that *B* matches *A* in semantic, written as $B \rightarrow A$.

Definition 3: For the concept set *A* and *B*, there always exists a concept *Bi* in the set *B* matches each concept *Ai* in the set *A*, it is called that set *B* matches set *A* in semantic, written as $B \rightarrow A$.

Based on the idea of ontology to describe the concept, an accurate description of the interfaces of the components has to be made. The inputs and outputs of the components are abstracted and ontology is used to describe them. This is convenient for matching the inputs and outputs to form the composition solution.

Definition 4 [9]: Use *COM (I, O)* to represent the internetware component. The *COM* is the name of component, *I* is the input set and *O* is the output set.

Definition 5: For the input parameter *A* of *COM1(I, O)*, if there exists an output parameter of *COM2(I, O)*, *A* and *B* meet the relationship of $B \rightarrow A$, called *COM2* is a precursor internetware component of A.

Definition 6: For an internetware component *COM(I, O)*; if its actual input parameter set is *A* and it meets the relationship of $A \rightarrow I$ then the *COM* has the conditions for execution. If an output parameter set of some internetware component or internetware component set itself includes *A*, this component or component set is called the precursor component of *COM (I, O)*.

It is easy to match the input and output interfaces of the internetware components based on the above description of the parameters. Then find out the precursors and the successors of each component are obtained and by this result, the needed components are combined and the composition solution to the users is given.

5. ALGORITHM DESIGN

First, the terms of the leaf nodes of the function subtrees in the internetware component trees are encoded by the terminology dictionary and leaf nodes are related to the corresponding strings. Secondly, the internetware components in the virtualized storage pool are retrieved by matching the function strings, the retrieved components are sorted by the matching degree and the results are shown to the user. Finally, the user selected components are combined automatically by the algorithm of figure backward iteration and the composition results are given to the users.

5.1. Make the Matching String

The terms of the leaf nodes of the function sub-tree are encoded in each component tree to simplify the description of the facets.

Input: The virtualized storage pool *I*.

Output: The string set SD of function.

(1) For each internetware component tree *T* in the virtualized storage pool *I*.

(2) Encode every term of the leaf node of the function sub- tree and add the code of every term to *S*.

(3) Sort the characters in *S* by the dictionary order.

(4) Add the root node of the tree *T* to the end of S.

(5) Add *S* into the set SD.

(6) End for

(7) Return SD

5.2. Match and Retrieve

The needed function strings are used to match each string in the component string set and the matching degree of the matched string are recorded after getting the internetware component string set. The retrieved strings are sorted by the matching degree. The retrieved and sorted components are obtained by using the retrieved strings and the results are given to the user.

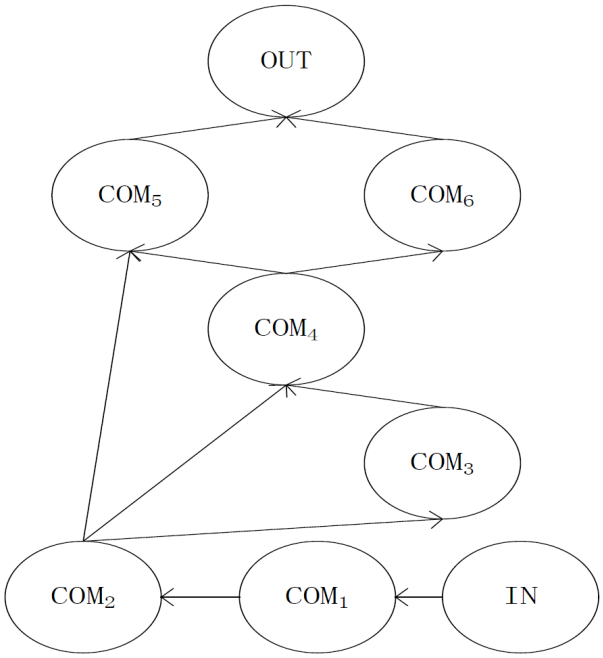

Fig. (4). Composition of the components.

Input: The needed function string *I* of the user and the string set SD of function

Output: The retrieved internetware set *M* sorted by the matching degree.

(1) For each string *S0* in the string set SD.

(2) For each character *c* in the string I.

(3) While the character *sc* in string *S0* is not equal to the character '@'

(4) If *sc*==*c*, the match number *mnum*=*mnum*+*1*, then break *i.*

(5) The character *sc* becomes the next character in string *S0*.

(6) End while

(7) The character *c* becomes the next character in string *I.*

(8) End for.

(9) Calculate the matching degree mm of S0, if mm!=0, then add the root node R0 and the matching degree mm of S0 to M

(10) Next string S1 in SD

(11) End for

(12) Sort the root node *R* in the *M* by the matching degree *mm*.

(13) Return *M*

5.3. Internetware Components Composition

In the process of composition, the first step is to match the input and output interfaces of each component to get the precursor set. The components that are not the precursors of others are obtained and these components are the output of the system. According to the output information and the precursor sets the algorithm of figure backward iteration are used [9, 14] to combine the components automatically and form the composition solution.

The following examples are used to explain the algorithm of figure backward iteration of the components composition: there are some internetware components as followed: COM1(Ai, Bi, Co), COM2(Ci, Do, Eo, Fo), COM3(Ei, Fo, Go), COM4(Di, Fi, Ho, Io), COM5(Di, Ii, Jo, Ko) and COM6(Hi, Lo). By matching the input and output interfaces of the components, the precursor set of each component are obtained which are as follows: COM1 = {}, COM2 = {COM1}, COM3 = {COM2}, COM4 = {COM2, COM3}, COM5 = {COM2, COM4}, COM6 = {COM4}.

According to the above precursor sets, it is easy to know that the component COM1 needs an external input and COM5 and COM6 are the system's output. After obtaining the relationships of the components, the algorithm of figure backward iteration is used to combine the components and obtain the composition solution. The Fig. (**4**) shows the composition of the components. In Fig. (**4**), OUT represents the system's external output, IN represents the external input required by the system and the arrows represents the order of internetware components execution.

The composition algorithm of the internetware components is described as follows:

Input: An internetware component set M selected from the retrieved components and an assumed external input set I

Output: The composition solution S of the internetware components

(1) For each component C in the internetware component set M.

(2) Set the precursor node set F of each component to NULL. Set a flag J and initialize J is 0. J is used for judging whether C has become the precursor of other components

(3) End for

(4) For each component C1 in M

(5) For each component C2 in M

(6) If C2 is the precursor of C1, add C2 to the precursor set F of C1 and set the flag J of C2 to 1

(7) End for

(8) If any input parameter of C1 is not matched, add the external input I to the precursor set F of C1

(9) End for

(10) For each component C in M

(11) If flag J of C is zero, add C to the output stack O

(12) End for

(13) While stack O is not NULL

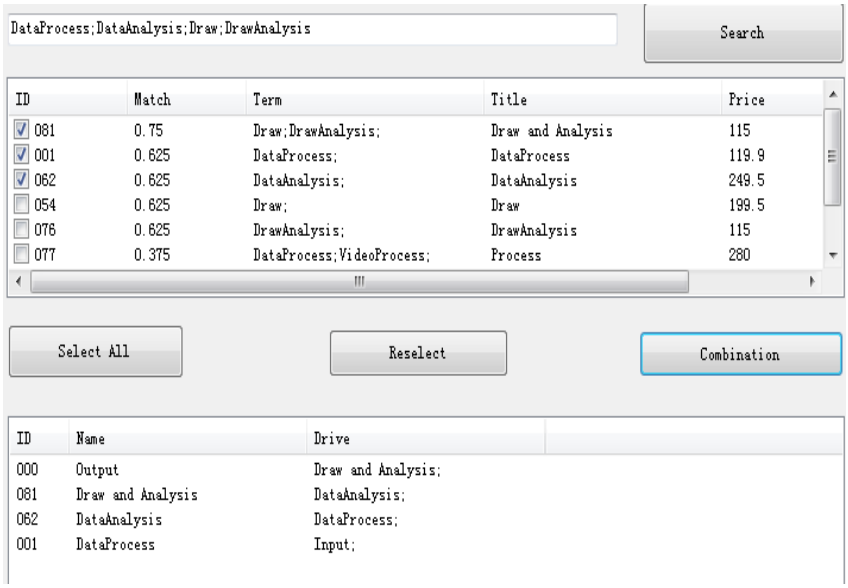

Fig. (5). Results of experiment.

(14) Get the internetware component C from the Stack O. Add C to S.

(15) Get the precursor node set of C and add the precursor node that's not the external input I to the stack O.

(16) End while

(17) Return the composition solution S.

6. EXPERIMENT

C# is used to implement this experiment in Microsoft Visual Studio 2008 and Microsoft SQL Server 2008. The internetware components in the virtualized storage pool are indexed, the needed components by the functions the users provided are retrieved and the retrieved components by the matching degree are sorted from high to low. The information of the sorted components are listed and finally the interfaces of components the users selected are describe by using ontology to make the description to unify and combine the components by using figure backward iteration automatically and the feedback to the users is given.

In (Fig. **5**), Output represents the external output of the combined internetware components, and Input represents the external input for the composition. The column of drive represents the required precursors of the internetware components. The experiment proves that the algorithm of component retrieval and composition is feasible.

CONCLUSION

XML used to describe that the facets of the internetware components has strong structure and it is easy to transform the description described by XML to a component tree. Using the idea of tree matching to retrieve the components, the matching degree of each retrieved component is calculated and the retrieved components are sorted for the users to find the required components. By using the concept of ontology to describe the interfaces of the components to unify the description in semantics, it is convenient to match the input and output interface by the process of composition and combine the selected components by using the algorithm of figure backward iteration automatically and return the feedback to the users.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work was supported by the Fundamental Research Funds for the Central Universities (Grand No. 2014-VII-027), the National Natural Science Foundation of China (Grand No. 51179146), and the Natural Science Foundation of Hubei Province (Grand No.2011CDB254).

REFERENCES

[1] H. Mei, and X. Liu, "Internetware: An emerging software paradigm for internet computing," *Journal of Computer Science and Technology*, vol. 26, no. 4, pp. 588-599, 2011.

[2] Y. Gu, P. Xu, and L. Xu, "A new software development methods under the network environment-Internetware," *Computer Knowledge and Technology*, vol. 7, no. 21, pp. 5141-5143, 2011.

[3] Y. Jia, and S. Zheng, "Software evolution based on service-oriented requirement in internetware," *4th International Conference on Computer Research and Development*, vol. 39, 2012.

[4] F. Yang, H. Mei, and J. Lu, "Some discussion on the development of software technology," *Acta Electronica Sinica*, vol. 30, no. 12A, pp. 1901-1906, 2002.

[5] Y. Xue, R. Xu, and L. Qian, "A summary of internetware-a new software modality under the internet computing environment," *Computer Engineering and Applications*,vol.15, no. 14, pp. 38-41, 2004.

[6] J. Lv, X. Ma,and X. Tao, "The research and development of the internetware, *Science in China Series E: Information Sciences,* vol. 36, no. 10, pp. 1037-1080, 2006.

[7] Y. Shu, Z. Chen, and X. Peng," A study of the component retrieving methods based on faceted classification," *Computer Engineering & Science*, vol. 32, no. 11, pp.156-160, 2010.

[8] J. Yue, and Z. Zhang, "A Survey on ontology specification languages," *Computer Science*, vol. 33, no. 2, pp. 158-162, 2006.

[9] H. Li, G. Hu, and Y. Yuan, "Research on web services composition based Ontology," *GIS*, pp. 164-169, 2006.

[10] Y. Yu, and Y. Jiang, "Improved method of tree matching using XML," *Computer Engineering and Applications*, vol. 48, no. 20, pp. 177-181, 2012.

[11] Q. Yao, X. Ding, and Z. Ran," Implementation and research of component retrieval algorithm based on XML and tree relaxation matching," *Application Research of Computers*, vol. 25, no. 4, pp. 1013-1019, 2008.

[12] Q. Yao, and B. Liu, "Research of efficient component retrieval algorithm based on facet classification," *Computer Engineering and Applications*, vol. 46, no. 2, pp. 118-120, 2010.

[13] X. Jia, D. Chen, and M. Yan, "Research on matching model and algorithm for faceted based software component query", *Journal of computer research and development*, vol. 41, no. 10, pp. 1634-1638, 2004.

[14] J. Liu, and H. Xu," The reverse iterator method of web services composition based on semantic ontology," *Science Technology and Engineering*, vol. 8, no. 19, pp. 5408-5423, 2008.

Received: September 22, 2014 Revised: November 03, 2014 Accepted: November 06, 2014