DE GRUYTER OPEN

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 1

Sofia • 2015

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0002

A Component Retrieval Tree Matching Algorithm Based on a Faceted Classification Scheme

Yao Wang[1], *Bo Wang*[2], *Minghan Liu*[1]

[1]*School of Software, Harbin University of Science and Technology, Harbin150040, Heilongjiang, China*
[2]*School of Automation, Harbin University of Science and Technology, Harbin150080, Heilongjiang, China*
Emails: wangyao1981@hrbust.edu.cn hust_wb@126.com 827555626@qq.com

Abstract: *An efficient scheme of component retrieval can significantly reduce the cost of software reuse. For this purpose, a method of successfully retrieving of specified components from the component repository is a crucial consideration. However, neither the retrieval efficiency, nor the query-matching rate of the traditional method, which is based on a faceted classification scheme, satisfies the requirements of component retrieval. In this paper a novel component retrieval method combining the features of the faceted classification scheme and the theory of tree matching is proposed. This method not only accurately retrieves components that match queries, but also considers any incomplete descriptions of the retrieval component to completely ensure the relaxation ability of the component retrieval. The experimental results show that the retrieval matching method proposed is highly efficient, and it retrieves feasibly and efficiently the components.*

Keywords: *Component repository, component retrieval, faceted classification scheme, software reuse, tree matching.*

1. Introduction

In recent years, alongside with the rapidly growing application of computer technologies within various fields, the scale and complexity of the software has also dramatically increased, leading to a phenomenon that many call "software crisis" [1]. In an effort to avoid this, researchers and developers have explored myriad methods and commercial processes [2-5]. Software reuse technology, which is characterized by the development of new systems using existing software

components, such as source codes, requirement specifications, designed structure, development tools, integration environments and data analysis, is a novel approach to address this issue [6]. When a software reuse system is implemented, a mass of reuse components accumulate, requiring a component repository to classify and manage them. Containing the correct components in the repository requires the support of automatic retrieval tools. The reuse of any component is characterized by retrieval, understanding and modification, which altogether form its reuse cost, calculated accordingly by the sum of the Retrieval Costs (RC), Understanding Costs (UC) and Modification Costs (MC). An efficient component retrieval scheme can greatly reduce RC and MC [7], while components classification is necessary simply to achieve the components retrieval efficiently and conveniently. Both component repository management and reuse cost reduction require thorough analysis and evaluation of the components classification and the retrieval schemes being utilized, which ultimately provide technical support for the reuse projects in practice.

The idea of software reuse has been recognized by the academic community. Mcilroy proposed the idea of software reuse at NATO Software Engineering Conference [8] and he also offered the idea of software development, based on components. This idea for software engineering has had a profound impact. How to quickly and accurately obtain the required components from the component library is the core issue of automation software component reuse. Podgurski and Pierce [9] proposed a component sampling retrieval for component behaviour representation. Amy Moormann proposed the signature match and specification match against formal component description [10]. For component descriptions which take the form of differences, other researchers have developed many component retrieval methods. Currently, depending on the complexity and retrieval results, the retrieval methods can be categorized as text-based (based on their morphological description), or based on three types of lexical regulations, either artificial intelligence, hypertext, or information science, according to components' expressed departures. These retrieval methods display distinct and sometimes quite advantageous characteristics, but are also limited in significant ways. Many previously proposed component classification and retrieval methods are uneven, or do not account for incomplete descriptions of the retrieved parts in addition to the recall rate, precision rate or retrieval efficiency. For example, Chung Yeonkyoung [11] developed a faceted classification that enables the conceptualization and the organization of Korean food information. The result of this study will be used for organizing, searching, retrieving, and providing Korean food information efficiently around the world. Pontes and Vieira [12] determined the value of application of a faceted taxonomy for the organization of the knowledge in a digital library of theses and dissertations. Liu and Wang [13] according to the current widely used methods described and the characteristics of the components faceted classification, starting from the component reuse by different search methods, he made two matching algorithms. Beida Jade Bird APTECH is to adopt faceted classification based multi-classification model, combining a method to classify the component description (see [14]). Zhang and

Lichao [15], Dong and Yafei [16], Feng and Liwei [17] also proposed some related component retrieval methods.

In this paper a novel component retrieval method is proposed, based on the features of a faceted classification scheme plus the tree-matching theory. Firstly, a faceted classification scheme is designed to form the structure of the retrieval model; next, the tree-matching method is used to solve the preceding model. The experimental results show that the method proposed not only accurately retrieves components that match queries, but also considers incomplete descriptions of the retrieval components to ensure the relaxation ability of component retrieval.

2. Faceted classification scheme of components

Typically, a faceted classification scheme is composed of facets that describe a set of essential characteristics of components [18]. Each facet is composed of a set of basic terminology, using different sides to place the respective components in the component repository under accurate classification, including application fields of the component, function, operation target and use of the environment, accounting the relationship between the class hierarchies (the class hierarchy is formed between each term's structured term spaces). Fig. 1 shows the basic structure of the component faceted classification.

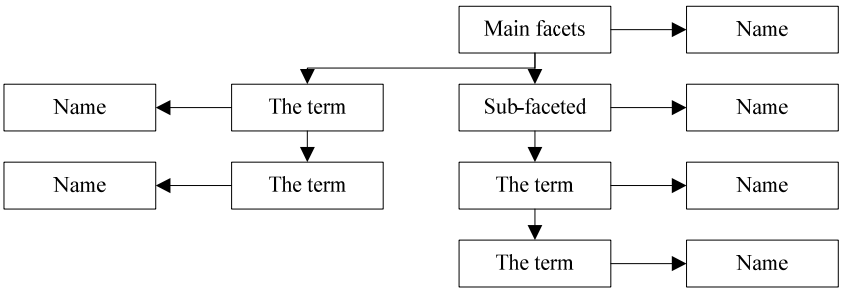


Fig. 1. Faceted classification schema for components

As shown in Fig. 1, one component can be described by each of the facets of one or more terms; however, each facet reflects a division of the components in the repository. Each component in the component repository has corresponding attributes, where its property value is equivalent to the term of the facet.

3. Component retrieval tree-matching model

This section introduces tree-mapping and the related theories applied to the component retrieval, inclusiveness and relaxation matching.

3.1. Tree mapping theory

Definition 1. Tree. Given an acyclic connected graph $T=(V, E, \text{root}(T))$, where V and E represent T's finite set of nodes and edges respectively, and root(T) is a

particular node in the graph, T is defined as a tree if it satisfies the following conditions:

(i) Node root(T) has no parent.

(ii) Except node root(T), other nodes in V have only one parent.

(iii) Edge set E is in a binary relation to V, and it is anti-reflexive, anti-symmetric and transitive.

(iv) $\forall v \in V$ and $v \neq$ root(T) $\exists$ (root(T), v) $\in E^+$, E^+ is the transitive closure of E.

Additionally, if the tree T has disordered sibling relationships, T is considered an unordered tree.

Definition 2. Tree Map. Given two unordered trees $T_1 = (V_1, E_1, \text{root}(T_1))$ and $T_2 = (V_2, E_2, \text{root}(T_2))$, a mapping from T_1 to T_2, denoted as f, can be defined as: $f \subseteq V_1 \times V_2$. Also, $\forall (v_{1i}, v_{2i}), (v_{1i}, v_{2i}) \in f$ has the following properties:

(i) $v_{1i}=v_{2i} \Leftrightarrow v_{1i}=v_{2i}$, where mapping f is a bisection from tree T_1 to T_2;

(ii) v_{1i}=ancestor(v_{1j}) $\Leftrightarrow$ v_{2i}=ancestor(v_{2j}), where mapping f maintains the relationship between the nodes, like ancestors and descendants; where y = ancestor(x), y is the offspring of x.

The mapping f domain is defined as Domain(f)=$\{v_l \in V_l | \exists\ v_2 \in V_2 : (v_1,\ v_2) \in f\} \subseteq V_1$; and the range is Range(f)=$\{v_2 \in V_2 | \exists\ v_1 \in V_1 : (v_1,\ v_2) \in f\} \subseteq V_2$.

3.2. A tree match model

Q is a search tree, and the subset of Q nodes in the cluster is Q_{sub}. T is a component describing a tree, a subset of T nodes in the cluster is T_{sub}, and the following definition of the tree match component retrieval model is valid.

Definition 3. Sub-tree match. If there is a mapping from Q_{sub} to T_{sub}, denoted as f, then f is a sub-tree match if it satisfies the following conditions:

(i) $v_1=v_2 \Leftrightarrow f(v_1) = f(v_2)$, $v_1, v_2 \in Q_{sub}$, $v_1, v_2 \in$Domain(f);

(ii) v_1=parent(v_2)$\Leftrightarrow$ $f(v_1)$ =parent($f(v_2)$), where y = parent(x), y is the parent of x;

(iii) in Q, all leaf nodes are leaf nodes of T.

Fig. 2 shows a sub-tree match model.

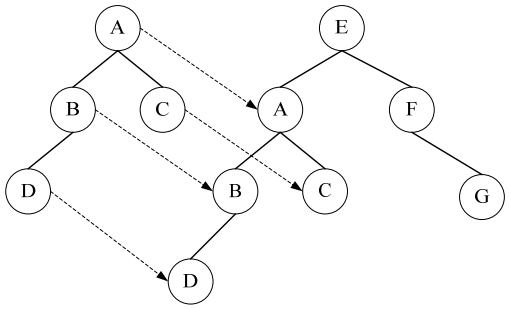


Fig. 2. A sub-tree match model

Definition 4. Contain Match. If there is a mapping Q_{sub} to T_{sub}, denoted as f, then f is a contain match, as long as it satisfies the following conditions:

(i) $v_1=v_2 \Leftrightarrow f(v_1)=f(v_2)$, v_1, $v_2 \in Q_{sub}$, v_1, $v_2 \in$Domain(f);

(ii) v_1=parent(v_2) $\Leftrightarrow f(v_1)$ =ancestor ($f(v_2)$);

(iii) In Q, all leaf nodes are leaf nodes of T.

Fig. 3 shows a contain match model.

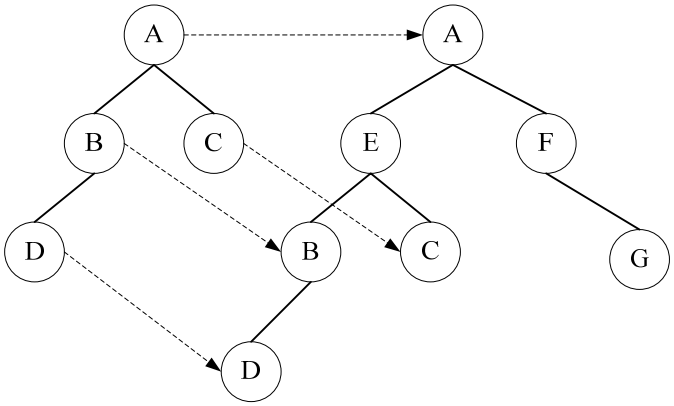


Fig. 3. A contain match model

Definition 5. Slack Match. If there is a mapping from Q_{sub} to T_{sub}, denoted as f, then f is a slack match if it satisfies the following conditions:

(i) $v_1=v_2 \Leftrightarrow f(v_1)=f(v_2)$, v_1, $v_2 \in Q_{sub}$, v_1, $v_2 \in$Domain(f);

(ii) v_1=parent(v_2) $\Leftrightarrow f(v_1)$ =ancestor($f(v_2)$);

(iii) In Q all leaf nodes are contained in T, but not necessarily leaf nodes of T.

Fig. 4 shows a slack match model.

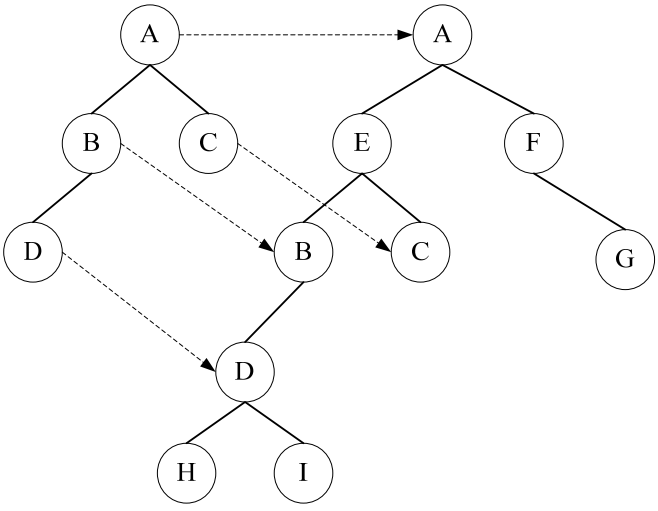


Fig. 4. A slack match model

4. Component retrieval tree matching algorithm

4.1. The term spatial encoding strategy

Definition 6. Term. A term is a word or phrase which is used to describe a fixed set of faceted attribute values. The encoding of each of the terms is referred to as term ID.

Definition 7. Term space. A structured set of legal terms is called a term space. When making a component retrieval, the component description tree and component retrieval tree terminology come from the term space. In order to facilitate the retrieval, the term must be encoded in the term space according to the general-special relationship. A specific coding strategy can be described as follows:

1) set the encoding to the terms of a unique ID, and then encode the first bit as a faceted ID, which represents the facets of components;

2) encode each bit representing a level term space, first the first layer, then its facets, then the first two represent the second layer, and so on;

3) arrange the code with the same superior terms, according to the dictionary sequence; every coding value indicates a relative position in terms of the hierarchical tree, within the range from A to Z.

4.2. Establishing of the component-tree index

The basic concept behind the generation node hierarchy ID method in the component tree is to first preorder the traversal component tree, according to the node terminology ID in the component tree to determine the relationship between the pairs of nodes in the term space – if the node has a father-son relationship, the child node is identified as “1”, otherwise as “0”. Next, according to the specific identification method, the level ID is in turn passed to the leaf nodes, which generates a hierarchical level ID for the leaf nodes that is then used to determine the match type. The algorithm is detailed as follows.

Algorithm 1. A Leaf Node Level String Generation Algorithm (LNLSG)
Input: Component tree *T*, the term ID of each node.
Output: Leaf node level string.
Steps:
Preorder the component tree *T*, generate a leaf node set *Leafset*;
$i = 1$;
While $Leafset \neq \emptyset$ **do**
$j = i + 1$;
End while
While $Leafset \neq \emptyset$ **do**
If t_i is the father of t_j term **then**
Return t_i level *ID* + "1";
Else if t_i is the ancestor of the term t_j **then**
Return t_i level *ID* + (*i*+ level hierarchy *j*) “0” + “1”;
j ++;
End while
i ++;
End while.

4.3. Description of the component matching algorithm

The component matching algorithm first uses LNLSG algorithm to generate the terminology string St1 and the level string St2 for the component retrieval tree, then makes the tree matching problem between the component description tree and the component retrieval tree into a terminology string, and a level string matching problem between the leaf nodes of these trees. A sorted set of components is then obtained, depending on the degree of match with respect to the match type. The specific algorithm is described as follows:

Algorithm 2. Component Matching Algorithm (CMA)

Input: Terminology string of the component description tree St1; terminology string of the component retrieval tree Sct1; level string of the component description tree St2; level string of the component retrieval tree Sct2.

Output: The set of components *C* sorted by the matching degree.

Steps:

Obtain the term ID from the term string Sct1, record it as ID_*t*;

If ID_*t* is included in the term string St1, then go to Step 4;

If ID_*t* is a prefix of the term ID in the term string St1, it is consistent with relaxation matching and the matching result will be stored in the collection C_3. Otherwise, go to Step 1;

If ID_*t* is equal to the corresponding level ID, then it is consistent with sub-tree matching, and the matching result will be stored in collection C_1; otherwise, it is consistent with the containing matching rules, and the match result will be stored in collection C_2. Go to Step 1;

Return the retrieved set of components $C= C_1+ C_2+ C_3$.

4.4. Complexity

The next step is to initialize the string phase component level, assuming the presence of members query tree *Q*, having *n* nodes and that there is a component description tree *T* with *m* nodes. Using LNLSG algorithm generates two components in the string tree hierarchy, where the complexity of the tree hierarchy *Q*'s generated string is $O(n + n^2)$, and the tree-level string generated *T*'s time complexity is $O(m + m^2)$. Thus, the time complexity of this stage is $O(m^2 + n^2)$. To perform member tree-type matching, using an algorithm (CMA), the members of the query tree *Q* terminology string, the string size, and the level are *m*, and the component description of the string tree *T* terminology, the string size and the level are *n*. The complexity of time during the term string matching algorithm is $O(mn)$. To calculate the string matching level, it is assumed that the member-level query tree *Q*'s maximum length of string elements is SCT_max, and that the component-level description of the largest element length string tree *T* is ST_max. The subtree matching algorithm time complexity is then O(SCT_max). As for the other two types, the time complexity of the algorithm is O(SCT_maxST_max). The resultant component set *C* of time complexity $O(m^2 + n^2 + mn$SCT.maxST_max) = O (mnSCT_max.ST_max).

5. Experimental results and analysis

The component repository used in our experiment contains more than 1000 components, including commercial components, purchasing components, and personality components developed by professionals. The experiment can be divided into two parts: the first one calculates the recall and precision ratios, while the second one verifies the retrieval efficiency.

5.1. Statistics of the recall ratio and precision ratio

The experiment was divided into two groups, A and B, with 10 people per a group. All participants had some basic knowledge of their respective component repositories, but lacked detailed understanding of the specific classification scheme for components. Each participant retrieved 30 components from their system. The components for Group A were retrieved using keywords (query results are keywords that match a queried component.) The components for Group B were retrieved using faceted search methods (the search results included components that both exactly matched and incompletely matched the query criteria). Fig. 5 shows the average component recall and the precision ratios for both groups.

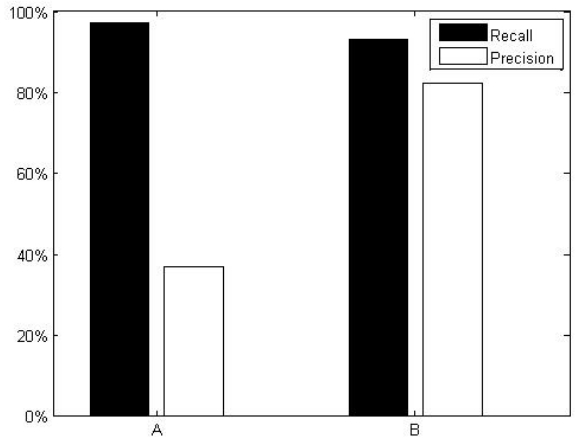


Fig. 5. Statistics of the recall and precision for the groups

Keyword searching, as shown in Fig. 5, had a high recall rate, but relatively low precision. Faceted retrieval methods showed an average precision rate of 83%, and average recall rate of 90%, suggesting that this method is highly applicable to component retrieval, and that keyword searching functions are best as a secondary aid to retrieval.

5.2. Retrieval efficiency statistics

The terms containing 5-25 component leaf nodes were selected according to retrieval conditions. The participants were given sub-tree match experiments containing both matches and slack matches. Fig. 6 shows statistics for the time needed to complete the matching.

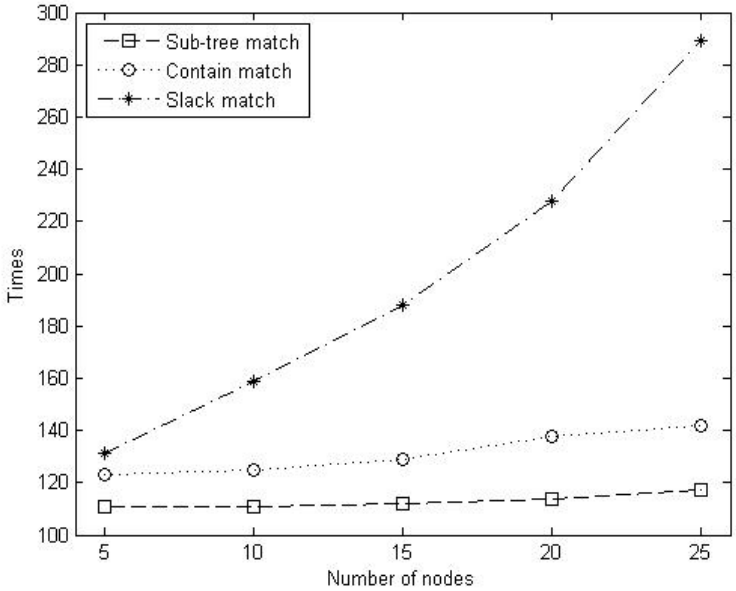


Fig. 6. Component retrieval efficiency

As shown in Fig. 6, determined by the complexity of the matching algorithm, the slack matches performed better. For each match, the query response time increased with the increase of the number of points in the retrieval tree leaf nodes, but even under conditions of relaxation matching, the largest amount of time required to complete the process was not more than 300 milliseconds (25 leaf nodes), indicating higher retrieval efficiency of the method proposed.

6. Conclusion

This paper has analyzed the characteristics of component retrieval and proposed a component retrieval method based on faceted classification plus tree-matching. The experimental results verified the proposed method's efficiency by demonstrating its high recall and precision rates. For this purpose, the method was proven feasible for application in practice.

The future research will further improve the matching model and will consider allowing the presence of matching faceted information that is staggered or decomposed, thereby improving the overall recall rate.

Acknowledgement: This work was supported by: National Natural Foundation of China, under Grant No 61172167; Natural Science Foundation of Heilongjiang Province of China under Grant No F201311; Foundation of Heilongjiang Educational Committee under Grant No 12531119. The authors would also like to express their deep appreciation to all anonymous reviewers for their kind comments.

References

1. Mili, H., F. Mili. Reusing Software: Issues and Research Directions. Software Engineering. – IEEE Transactions on, Vol. **21**, 1995, No 6, pp. 528-562.
2. Layzell, P. Addressing the Software Evolution Crisis through a Service-Oriented View of Software: A Roadmap for Software Engineering and Maintenance Research. – In: Proc. of IEEE International Conference on Software Maintenance, 2001, pp. 5-9.
3. Mandal, A. A Model for Modern Software Development Process to Cater the Present Software. 2009. IACC 2009. – In: Proc. of IEEE International on Advance Computing, 2009, pp. 1617-1623.
4. Li, H., J. van Katwijk. Issues Concerning Software Reuse-in-the-Large. – In: Proc. of Second International Conference on Systems Integration, 1992, pp. 66-75.
5. Smyrniotis, C. Rapid Prototyping: A Cure for Software Crisis, 1990. – In: Proc. of the Twenty-Third Annual Hawaii International on System Sciences, Vol. **2**, 1990, pp. 202-210.
6. Lee, N. Y., C. R. Litecky. An Empirical Study of Software Reuse with Special Attention to Ada. –IEEE Transactions on Software Engineering, Vol. **23**, 1997, No 9, pp. 537-549.
7. Chang, Jichuan, Li Keqin. Bluebird Systems Reusable Software Components Represent and Query. – Journal of Software, Vol. **28**, 2000, No 8, pp. 20-24.
8. Mili, H. Reuse Based Software Engineering. New York, John Wiley&Sons, Inc., 2002, pp. 444-459.
9. Podgurski, A., L. Pierce. Retrieving Reusable Software by Sampling Behavior. – ACM Transactions on Software Engineering and Methodology, Vol. **2**, 1993, No 3, pp. 286-303.
10. Zaremski, A. M. Signature and Specification Matching. Ph. D. Thesis, School of Computer Science Carnegie Mellon University, 1996.
11. Chung, Yeonkyoung. A Study on Structure of a Faceted Classification for Organizing Korean Food Information. – Journal of Korean Library and Information Science Society, Vol. **47**, 2013, No 1, pp. 15-37.
12. Pontes, Flavio Vieira. Knowledge Organization in Digital Environments: Faceted Classification Theory Applied. – Perspectivas em Ciência da Informação, Vol. **17**, 2012, No 4, pp. 18-40.
13. Liu, Y. P., J. P. Wang. Research on Component Retrieval Method Based on Faceted Classification Described. – Applied Mechanics and Materials, Vol. **121-127**, 2012, pp. 1727-1733.
14. Sindre, G., R. Conradi, E. Karlson. The REBOOT Approach to Software Reuse. – Journal of Systems and Software, Vol. **30**, 1995, No 3, pp. 201-212.
15. Zhang, Lei, Chen Lichao. Study on Tags Representation of Components and Tags Based Components Retrieval. – Journal of Chinese Computer Systems, Vol. **34**, 2013, No 5, pp. 1077-1079.
16. Dong, Yuehua, Ma Yafei. Research on Component Retrieval Algorithm with Facet-Weight. – Computer Applications and Software, Vol. **30**, 2013, No 3, pp. 175-177.
17. Feng, Jing, Huang Liwei. Research on Classification and Optimal Query Method of Software Configuration Component. – Computer Engineering, Vol. **38**, 2012, No 1, pp. 63-67.
18. Qing, Gao, Zhang Xiaoming. Implement of Faceted Classification Model Based on Achieve Domain Component Library System. – Computer Engineering and Applications, Vol. **30**, 2003, pp. 82-87.