Research on Component Query Matching Algorithm for Component Library Based on Facet Description

Yue Li [a], Xianjiu Guo [b]

Dalian Ocean University, Dalian, China

[a]33977300@qq.com, [b]879753691@qq.com

Keywords: software reuse; component library; component retrieval.

Abstract. More excellent containment matching model is proposed based on component inclusion matching model. Moreover, matching cost algorithm is analyzed to get inclusive matching model. Therefore, efficiency of recall and precision of component are significantly improved comparing with traditional component query situation.

Introduction

Representation and retrieval of component are the two main core technologies of reusable software component library. Representation of component facet and corresponding retrieval technology has been widely applied. Among them, both REBOOT and NATO put forward classification scheme of reusable software component.

In addition, retrieval of component library takes into account incomplete description of component need to query. Query matching should have a certain degree of flexibility, not only can give the user returns but also corresponding matching degree, to provide useful information for the users to reuse components. At the same time, classification scheme of facet in each component library may be completely different. Therefore, the user may need to span multiple component libraries to find appropriate component. How to realize the retrieval component across the component library is an urgent problem to be solved [1-2].

Facet Description of Component and Query Representation

Tree Representation of Component Facet. With the application of component library based on network, XML has been described as a component mark-up language. XML document of a component can be mapped to an unordered labelled tree. At present, classification describing model is adopted on most of component facet. Figure 1 showed a facet document based on XML component facet and its tree representation.

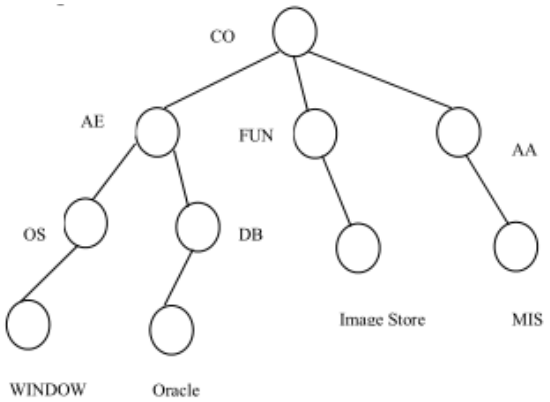


Fig.1 Facet Document and Tree Representation of Component

Component Query Representation. Component query can be expressed as a query tree, i.e., names of facet and sub-facet are turned into corresponding nodes and their sub-nodes, facet terms value needed to be inquired is turned into leaf nodes, and a virtual root node is used to combine them into a the query tree, as shown in figure 2.

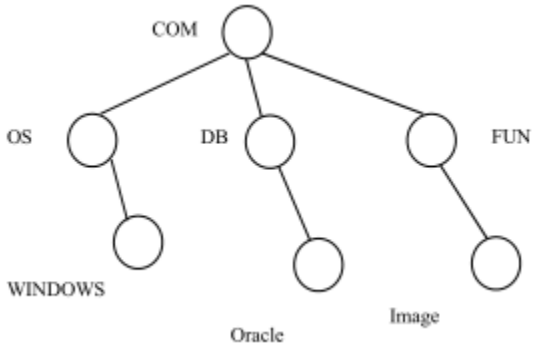


Fig.2 Component Query Tree

Therefore, component query is turned into the matching between query trees and facet description tree in the library. Many research results have been got on tree matching problems. Tree embedding or tree inclusion are directly related to the matching, that is, the best embedding or contain position of one tree in another is found.

Matching Model of Component Query

In order to analyze and states matching model of this paper, definition of tree inclusion is introduced, and tolerable definition is given. The meaning of component retrieval and matching model is analyzed combing with the features of component query.

Definition 1 Tree Inclusion / Tree Embedding

Suppose Q= (V, E, root (Q)) and D= (W, F, root (D)) are two unordered labelled tree. If there is a mapping f $V \rightarrow W$, which satisfies the following conditions:

(1)$u=v \Leftrightarrow f(u)=f(v)$, $u,v \in Domain(f)=V$

(2)label(v)=label(f(v))

(3)$u=parent(v) \Leftrightarrow f(u)=ancestor(f(v))$

We say Q is a included tree, while D contains Q.

If the above condition (3) is changed into u=parent (v) ⇔ f (u) =parent (f (v)), there is a embedding from Q to D.

Definition 2 Inclusion Matching

Suppose Q= (V, E, root (Q)) and D= (W, F, root (D)) are two unordered labelled tree V 'V and W' W. If for all u, $v \in V'$, there is a mapping f $V' \rightarrow W'$ meeting the following conditions:

(1)$u=v \Rightarrow f(u)=f(v)$, $u,v \in Domain(f)$

(2)$label(u) \approx label(f(u))$

(3)$u=ancestor(v) \Rightarrow f(u)=ancestor(f(v))$

Then f is called a tree inclusion matching from Q to D, referred to as the tree inclusion or inclusion matching.

Inclusion matching is shown in Figure 3 (a). Comparing with the definition of tree containment or tree embedding, it breaks through domain constraints of f and allows redundant query information, which fundamentally creates condition for component query recall, and only requires ancestor-descendant relationship. Therefore, it allows missing layer of participating node information for matching to improve recall ratio.

Definition 3 Containment Matching

Q= (V, E, root (Q)) and D= (W, F, root (D)) are two unordered labelled tree, $V' V$ and $W' W$. If for all u, $v \in V'$, there exists a mapping f from V' to W ' meeting the following conditions:

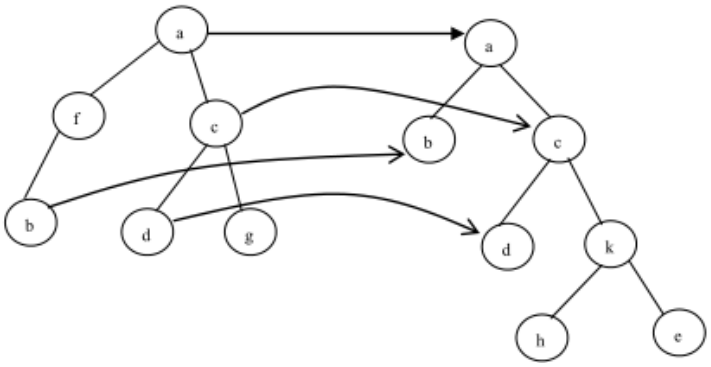
(1)$u=v \Rightarrow f(u)=f(v)$,$u,v \in Domain(f)$

(2)$label(u) \approx label(f(u))$

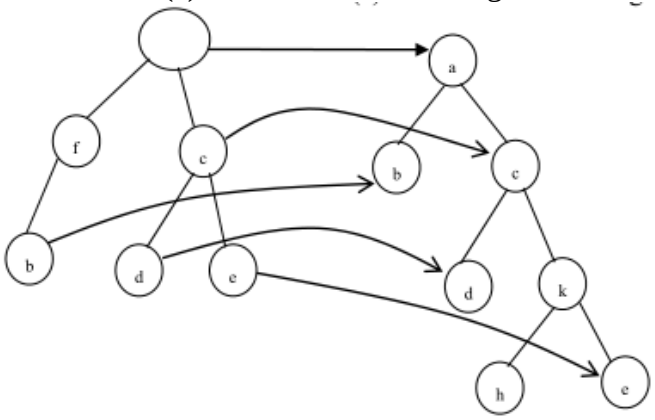
(3)$u=ancestor(v) \Rightarrow f(u)=ancestor(f(v))$

Then f is called a tolerable matching from Q to D, referred to as the tree tolerant or containment matching. Tolerable matching is a special case of inclusion matching. As shown in Figure 3 (b),

tolerance matching has higher requirements comparing with inclusion matching, and all term conditions of facet in query tree must be met. Without affecting component query efficiency, the precision is improved, and the user can query different component libraries with different classification schemes.



(a) Containment Matching



(b) Tolerable Matching

Fig. 3 Schematic Diagram of Two Tree Matching

Matching Algorithm and Analysis of Component Retrieval

Matching Cost of Component Retrieval. Based on above matching model, tree matching can be considered from the transformation between them. The main idea of tree transformation is to define some editing operations in advance, through which a tree T1 can be converted to another tree T2. Giving a real number for each edit operation $x \rightarrow y$, $x, y \in \{label\} \cup \{\varphi\}$ calls edit cost of this operation, denoted by $\gamma(x \rightarrow y)$. Essence of matching is a map. In order to define matching cost meeting the requirement of component matching features, concept of spectrum is introduced [3].

Algorithm Analysis of Matching Cost. The following algorithm can be used to a complete query.

Input: connection query tree array SearchTree [] and component array ComTree []

Output: component set R meeting the need of query

R=φ;

For each tree Q in query array

For each tree D in component array

If mismatching exists in any leaf node

break;

else

calculate matching cost by calling the calculating method of above matching cost(for non-leaf nodes)

end if

put component D to R;

end for

end for

Conclusion

This paper put forward an improved matching method for query in multiple component libraries based on containment matching model. Although the efficiency is a little lower than traditional model, the recall and precision ratio are enhanced. The research results of this paper can be widely used in component query based on facet description in multiple component libraries and network component library. In addition, component matching algorithm composed in this paper can be used as one of the effective methods for component retrieval.

Acknowledgement

The paper is funded by Science and Technology Department of Liaoning Province (No.2012216012) and National Marine Public Welfare Projects (No.201205023-4).

References

[1]Podgurski A,Pierce L.Retrieving Reusable Software by Sampling Behavior[J].ACM Transactions on Software Engineering and Methodology,1993,2(3):286-303.

[2]Zaremski A M.Signature and Specification Matching [D].School of Computer Science, Carnegie Mellon University, 1996.

[3]Merkl D,Tjoa A M,Kappel G.Learning the Semantic Similarity of Reusable Software Components[C].Proceedings of the 3rd International Conference on Software Reuse,1994:33-41.

[4]Zhang K Z.On the Editing Distance between Unordered Labeled Trees [J].Information Processing Letters, 1992, 42(3):133-139.