Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 327 – 332 Sept. 2007 ISSN 1003—7985

Retrieving reuse component based on semantic

Wang Yan Chen Ming Zhao Jianhui

(Department of Computer Science and Technology, China University of Petroleum, Beijing 102249, China)

Abstract: According to the current research status of component retrieval, the component description model based on facet classification is improved by adding semantic features. Furthermore, the component retrieval process model is put forward by combining the domain ontology with the relative concept match algorithm. A detailed illustration of a component reasoning engine and a component classification engine is given and the component classification algorithm is provided by using the Naive Bayes algorithm based on domain ontology. The experimental results show that the recall ratio and the precision ratio are obviously improved by using the method based on semantics, and demonstrate the feasibility and effectiveness of the proposed method.

Key words: domain ontology; facet classification; naïve Bayes algorithm; component matching; component-based software development

1 Background

In recent years, component-based software development (CBSD) has already become a research hotspot in the software engineering domain. With software reuse development and component repository scale expansion, how to enhance retrieval efficiency has become a key technical issue. The current component retrieval method is mainly divided into four types: the specification matching method, the artificial intelligence method, the hypertext method and the information science method[1-2].

1.1 Component retrieval based on specification matching

According to the actual form, specification matching is mainly divided into signature matching[3] and behavioral matching[4]. It can integrally describe the component computation semantics, and the accuracy of the retrieval is higher than that of the non-formalized method. But the description of the specification retrieval is more complex and the response time depends more on the adopted method. So at present this method has not been used widely.

1.2 Component retrieval based on artificial intelligence

This method can be divided into component retrieval based on sampling behavior[5], component retrieval based on knowledge library and component retrieval based on the artificial neural network. The component retrieval based on sampling behavior is only suitable for the code level component and mainly used in the specific library whose sample is relatively simple. The component retrieval based on the knowledge library is more accurate and efficient than the retrieval method based on key words, but the knowledge library is usually realized by manual methods, so its costs are very high. The component retrieval based on the artificial neural network requires the invariable dimension of the input vector. If the dimension changed, the algorithm would be redesigned. So this method depends more on artificial neural network technology.

1.3 Component retrieval based on hypertext

This method adopts hypertext-linking technology, so it is mainly suitable for linear retrieval. It requires that the user manipulate the browsing process. It is difficult for the more complex component repositories and it is only used for auxiliary retrieval.

1.4 Component retrieval based on information science

The component retrieval based on information science is more successful in practical application. The common uses of this method are enumerations, facet, the attribute value and the key word. Because of its simple realization and rich expression, the method[6] based on facets is paid more attention to. The component classification method based on facets is proposed by Reboot and NATO. But this method cannot describe the component semantics feature efficiently; therefore, it may bring little influence on the recall ratio and precision ratio of the retrieval.

2 Ontology Model in Specific Domain

Ontology is a conceptual model that can describe

Received 2007-05-18.

Foundation item: The National Natural Science Foundation of China (No. 60072006).

Biographies: Wang Yan (1974—), female, lecturer, wyhbsh@yahoo.com.cn; Chen Ming (1949—), male, professor, chenming@cup.edu.cn.

the system at the semantic and the knowledge levels. The purpose of this method is to gain the knowledge of the domain in a universal way. It provides the consistent understanding[7] to the concept in the domain, and strengthens the computer handling ability of the knowledge. Ontology has been widely applied in different fields of retrieval.

At present there are many methods of constructing the ontology model, for example skeletal methodology, Tove (Toronto virtual enterprise) and so on, but the ontology construction has not formed a normative criterion. It is good just playing well under its research environment. The main goal of this paper is to satisfy the demands of component retrieval, so we synthesize the existing construction ontology method and simplify the existing component domain ontology method.

Definition 1 (domain ontology) Using 3-tuple (C, A, R), where C represents the concept set in the domain, A represents the attribute set in the domain, and R represents the domain ontology element relation set. $R_p \in R$, R_p represents the mapping relationships among concepts and the key words; $R_s \in R$, R_s represents synonymy relationships among the key words, $R_v \in R$, R_v represents the part of relationships among concepts; $R_i \in R$, R_i represents the concept level relations.

We create the domain ontology model by using the concept chart and describe the domain ontology by using OWL (web ontology language) class, attribute, class axiom and so on. We also use the ontology edition tool (for example the quite famous Protégé tool) in the graphical user interface, and then automatically produce the OWL language description.

3 Component Description Model Based on Semantic

The existing component description method has given a more exhaustive consideration to the static feature of the component and has obtained a good solution, but it does not provide an effective method for dynamic feature description, while the dynamic feature is precisely one of the most important reasons that the retrieval quality (recall ratio, precision ratio and retrieval capability) is affected. In order to solve this problem, we combine the interface description with the domain ontology. The interface realizes one service function; it is connected with the concept that expresses this service function in the domain ontology. The input and output data of the interface are all described by the corresponding concepts in the domain ontology. So we can annotate the semantics of the interface service by using the domain ontology concept. By using this component description method, the service semantics provided by the component can be understood better on a higher level, and it also supports the retrieval, the matching and confirmation of the component effectively.

Definition 2 (component) The component is defined as the software unit that can be reused, and it has semantic integrity and correct grammar. The component is the complex of the basic information (C_BaseInformation), component facet (C_Facet), component interface (C_Interface) and component entity (C_Entity).

Component: : = 〈C_BaseInformation, C_Facet, C_Interface, C_Entity〉.

Definition 3 Component basic information (C_BaseInformation) mainly refers to the static information of the component, it includes the following parts: component ID, component name, component author, component function, component version, component size and component issue date. C_BaseInformation: : = 〈C_ID, C_Name, C_Author, C_Function, C_Version, C_Size, C_Date〉.

Definition 4 Component facet refers to the context that understands the component. Each facet is composed of a group of terms. The component facet is composed of component function, application environment, component type and operated object. C_Facet: : = 〈C_Function, Application_Environment, C_Type, Operated_Object〉. Where the component function refers to the service and function provided in the original software system, the application environment refers to the hardware or the software platform where the component is used; the component type refers to different stages of abstract levels in the software development history; the operated object refers to the object that the component operates (Generally it refers to the entity object that is involved in the domain).

Definition 5 (component interface) The component interface is the medium through which the component can exchange information with other components or the external environment. It includes the service request interface and the service provision interface. The interface is composed of interface name, interface function and interface semantic. C_Interface: : = 〈Interface_Name, Interface_Method, Interface_Semantic〉.

Definition 6 The interface method is composed of method name, method function and method parameter. Interface_Method: : = 〈Method_Name, Method_Function, Method_Parameter〉.

Definition 7 (interface semantic) Interface semantic can be divided into four parts, which are interface function, interface operation, interface parameter

and interface relationship. Interface_Semantic: : = 〈Interface_Function, Interface_Operation, Interface_Parameter, Interface_Relationship〉. Where the interface function description corresponds to the service function in the domain ontology; the operation corresponds to the concept of action in domain ontology; and the parameter description corresponds to the information resource in the domain ontology.

Definition 8 The interface relationship refers to the relationship between the components. It mainly includes version relationship, cooperation relationship, refinement relationship, inclusion relationship and dependence relationship. Interface_Relationship: : = 〈Version_Relationship, Cooperation_Relationship, Refinement_Relationship, Inclusion_Relationship, Dependence_Relationship〉. Where the version relationship refers to the relationships among the versions that the component has evolved, cooperation relationship refers to the relationships among various components that mutually cooperate and complete a task together; refinement relationship refers to the relationships among the components on the neighboring stage of the software life cycle; inclusion relationship refers to the containing relationships among different forms of components; dependence relationship is one kind of using relations, which means the change of a component will affect the use of other components.

Definition 9 Component entity refers to the instance that satisfies the interface specification and the semantic description. The concrete expression of the component entity is as follows:

C_Entity: : = 〈Index, Implementation_Body〉

4 Model of Component Retrieval Process

4.1 Ontology reasoning engine and component classification engine

The domain ontology reasoning engine uses the basic algorithm of mapping the key word for a concept. It namely finds the key word in the domain ontology, then maps the relation for a concept according to the relationships among the key words. For those concepts which have the part of relation, the domain ontology reasoning engine outputs the whole concept instead of the partial concept and replaces the concrete level concept by the more abstract one if there is a level relationship among the concepts.

The component description consults the component description method[8] from the component retrieval based on facet, the feature information is described as $D = \{T_1, T_2, T_3, \ldots, T_n\}$, where T_n is a domain feature of the component, and it is also an element in the domain ontology. The ontology reasoning engine transforms T_n into concept W_i by using relationships in the domain ontology. Then $D = \{W_1, W_2, W_3, \ldots, W_i\}$.

The component classification engine divides the component into different categories according to the component classification algorithm, then creates a component index library for the component repository according to this classification, namely each category is connected with an index library, the table in the index library records all component IDs. When retrieving a kind of component, the system directly finds the component storage position in the index library according to the component ID, so it can enhance the component retrieval time of response.

The component classification algorithm uses the naïve Bayes algorithm, the component domain feature information is described as $D = \{W_1, W_2, W_3, \ldots, W_i\}$; the domain feature W_i is substituted into the naïve Bayes algorithm, and then obtains the following equation:

$$V_{\mathrm{NB}} = \arg\max P(C_j) \prod_i P\left(\frac{w_i}{C_j}\right)$$

where $P(C_j)$ indicates the frequency of the component category C in training data; $P(w_i/C_j)$ indicates the frequency of the domain feature w_i which appears in component category C_j; V_{NB} represents the target value of the naïve Bayes classification. This target value determinates the category which the component belongs to. In the actual classification process, we use the following equation to estimate $P(w_i/C_j)$ for avoiding the situation of $P(w_i/C_j) = 0$.

$$P\left(\frac{w_i}{C_j}\right) = \frac{n_k - 1}{n + |\mathrm{Vocabulary}|}$$

where n expresses the total feature number that appears in this category; n_k indicates times the feature w_i appears; $|\mathrm{Vocabulary}|$ expresses the total feature number in the training set.

4.2 Component retrieval and matching

Component retrieval and matching is one of the key techniques for accomplishing software reuse. Along with the component repository scale expansion, how to find the appropriate component in the repository is a difficult problem. This paper constructs an intelligent component retrieval model based on the facet classification method, and adopts a user interface based on the natural language inquiry. The retrieval process is given in Fig. 1.

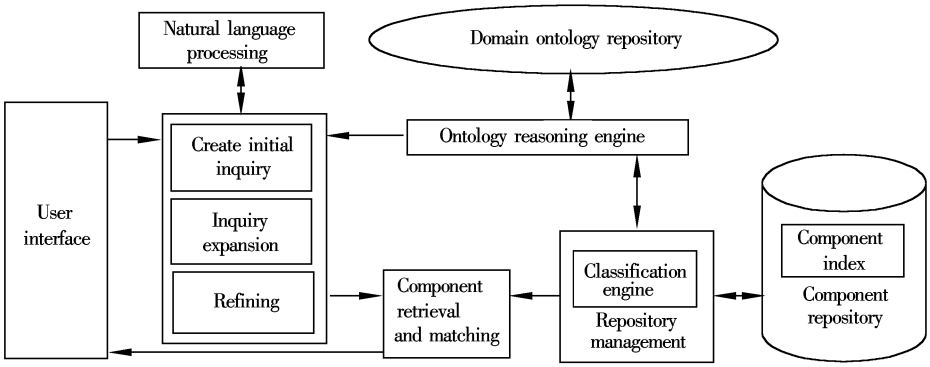

Fig. 1 Component retrieval model based on semantics

The retrieval method contains the following four steps:

① Creating the initial inquiry: The user puts forward the retrieval demand based on natural language, then creates the user's initial inquiry by using the natural language analysis, namely, transforming the user inquiry into the concept chart which expressed in the OWL language. This step involves natural language understanding and transforming the natural language to OWL language technology. This function consults the learning method based on cases[9]. First, a case library is built and the library stores the inquiry sentence and the corresponding OWL case set that is transformed from the inquiry. Secondly, finding the similar inquiry sentence in the case library, the new inquired sentence can be transformed into the corresponding OWL case. This case library is small in its initial period, while the natural sentence analysis quality has the direct relation to the case library scale. Therefore, it is necessary to expand the case library gradually. Fig. 2 shows the natural language query based on case learning.

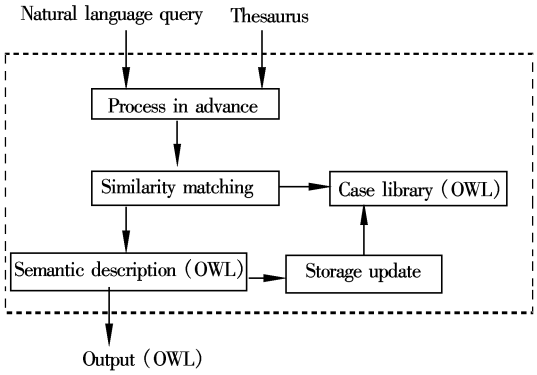

Fig. 2 Natural language query based on case learning

② The inquiry expansion and refining: First, the concept and the key words (which are used in the initial inquiry chart and are relative to the service function) are matched with the concept in the domain ontology; if the semantic similar degree satisfies the threshold value, the concept will be taken as the related concept and returned to the user for expanding the inquiry. In the inquiry expansion process, the user determines the desired domain ontology service function and its relationship with other concepts, then it establishes a retrieval context, refines the initial inquiry and produces the user's final inquiry.

③ Retrieving the desired component: In the above process, we expanded the correlation service function by using the semantic match according to the user's inquiry and obtained the final inquiry condition of the component which is expressed in the OWL language. The final inquiry condition is divided into the function demand and the non-function demand (for example language, running environment and so on). It then retrieves from the interface list according to the function demand. The percentage of the interface satisfying function demand expresses the behavior correlation degree. If the behavior correlation degree of the interface set is higher than the threshold value, the interface set sorted by its behavior correlation degree will be returned to the user. The user determines the component interface that matches his need. According to the inquiry's non-function demand and the interface, the system retrieves the corresponding component. If the desired component exists, the user may sort and choose the component according to other attributes (for example running rate, security rank), otherwise the user may retrieve again by the offered relative interface. The user is submitted not only the retrieved component basic information, but also the component assembly pattern formed by the relations (such as call, use) with the other interfaces.

④ User feedback: The user decides the usefulness of these components and provides the feedback, then adjusts the related threshold value according to these feedbacks.

5 Concept Matching Algorithm Based on Semantic

In the process of the inquiry expansion, the service function is properly expanded through the semantic matching. Namely, it provides other related functions

to the user according to the semantically similar algorithm. Its algorithm description is as follows.

Definition 10 The semantics distance between the concept Distance (C_1, C_2) represents the sum value of the n side weight on the shortest path semantic tree which is given by the following equation:

$$\text{Distance}(c_1, c_2) = \sum_{i=1}^{n} \text{Weight}_i$$

where Weight_i indicates the weight of the number i side on the shortest path that connects C_1 and C_2.

Because the domain ontology is stored by the upside down tree structure, according to the subjective judgment, the similar degree of the concept far from the root in the level tree is greater than that of the concept near to the root. Therefore, the concept depth in a tree is an important factor that should be considered. Different depth sides in the tree should be evaluated different weight values.

Definition 11 The depth degree of concept C in the tree Depth(C) refers to the side number n containing in the shortest path between the concept and the tree root. Depth$(C) = n$, where n is the side number of the shortest path.

Definition 12 Width degree between the concepts; Width(C) refers to the children node number under the same depth.

Definition 13 (weight value of concept C, Weight (C)) Because the side drawn out from concept C has an equal weight value, we stipulate that Weight(C) refers to the weight value of the side that is drawn out from concept C. It has the inverse ratio with the concept depth degree, and has the inverse ratio with the width degree of the concept in the same depth. The equation is as follows:

$$\text{Weight}(C) = \frac{1}{\text{Width}(C)} \frac{1}{\text{Depth}(C) + 1}$$

According to the above definitions, we have made the convert equation of the semantic similar degree and the semantic distance:

$$\text{Sim}(c_1, c_2) = 1 - \alpha\sqrt{\text{Distance}(c_1, c_2)}$$

Parameter $\alpha \in (0, 1)$ may be determined through many times of experiments, $\text{Sim}(c_1, c_2) \in (0, 1)$.

The similar degree between the two concept sets is as follows:

Supposed that $M = (m_1, m_2, \ldots, m_x)$, $N = (n_1, n_2, \ldots, n_y)$ are two concept sets, SimofSet(M, N) expresses the similar degree between the two concepts, and the equation is

$$\text{SimofSet}(M, N) = \sum_{i=1}^{x} \sum_{j=1}^{y} \text{Sim}(m_i, n_j)$$

When retrieving the interface, the user inputs the concept or the concept set in view of some feature, using the above equations to calculate the concept semantic similar degree in the interface list. The user can choose the proper one according to the sorted component. Then the user obtains the desired component through the map between the interface and the component entity.

6 Experimental Results

In order to validate the feasibility of the component retrieval based on semantics (CRBS), we test the existing teacher appraised component repository (TACR) by two retrieval methods: CRBS and component retrieval based on facet (CRBF). TACR has 150 actual components and 240 virtual components. By inputting different functions, the tester can obtain different data about the precision ratio and the recall ratio. Tab. 1 shows the experimental results.

Tab. 1 Experimental results of CRBS and CRBF

Category function	Desired component		Undesired component		Missed component		Precision ratio/%		Recall ratio/%	
	CRBF	CRBS	CRBF	CRBS	CRBF	CRBS	CRBF	CRBS	CRBF	CRBS
Database operation	8	9	1	1	1	0	87.5	88.9	87.5	88.9
Text format transformation	6	8	0	0	2	0	100	100	75	100
Template management	7	7	1	1	0	0	85.7	85.7	85.7	85.7
Preview evaluate template	6	7	0	0	1	0	100	100	85.7	85.7
Establishes new questionnaire	10	12	1	1	3	1	90	91.7	75	91.7
Questionnaire management	12	13	0	0	1	0	100	100	92.3	100
Questionnaire evaluate	9	9	0	0	0	0	100	100	100	100
Template frame management	10	11	0	0	1	0	100	100	90.9	100
Total	68	76	3	3	9	1	95.6	98.6	87.8	98.6

Definition 14 Precision ratio refers to the ratio of the desired retrieval components to the total retrieval components.

Definition 15 Recall ratio refers to the ratio of the desired retrieval components to the total related components in the component repository.

From Tab. 1 we conclude that the component retrieval based on semantics is obviously higher than the component retrieval based on the facets in the recall ratio, and it is slightly higher than the component retrieval based on the facets in the precision ratio.

7 Conclusion

In this paper, a component retrieval model based on semantics is put forward. We use the proper matching algorithm between the user demand and the component entity and enhance the component precision ratio and the recall ratio. In the experiment, the naïve Bayes algorithm requires a larger training set as the component quantity is not enough in TACR; therefore, we add some virtual components to the training set. The main characteristic of this system is to create a classification engine by combining the naïve Bayes algorithm with the domain ontology and introducing the intellectual processing method in the component retrieval. The future work includes: ① Improving the domain ontology repository and accurately mapping to the component features of this domain; ② Enhancing the response time of the component retrieval.

References

[1] Frakes W B, Pole T P. An empirical study of representation methods for reusable software components [J]. *IEEE Transactions on Software Engineering*, 1994, **20**(8): 617 - 630.

[2] Mili H, Rada R, Wang W, et al. Practitioner and SoftClass: a comparative study of two software reuse research projects [J]. *Systems and Software*, 1994, **27**(5): 147 - 170.

[3] Zaremski A M, Wing J M. Signature matching: a tool for using software libraries [J]. *ACM Transactions on Software Engineering and Methodology*, 1995, **4**(2): 156 - 180.

[4] Zaremski A M, Wing J M. Specification matching for software components [J]. *ACM Transactions on Software Engineering and Methodology*, 1997, **6**(4): 333 - 369.

[5] Podgurski A, Pierce L. Retrieving reusable software by sampling behavior [J]. *ACM Transactions on Software Engineering and Methodology*, 1993, **2**(3): 286 - 303.

[6] Zhang Yong, Zhu Sanyuan, Qian Leqiu, et al. A matching model for software component classified in faceted scheme [J]. *Journal of Software*, 2003, **14**(3): 401 - 408. (in Chinese)

[7] Chen Gang, Lu Ruqian, Jin Zhi. Constructing virtual domain ontologies based on domain knowledge reuse [J]. *Journal of Software*, 2003, **14**(3): 350 - 355. (in Chinese)

[8] Gao Qiang, Zhang Xiaoming, Bian Xiaofan. The study of faceted classification scheme of the specific domain-based RSL[J]. *Computer Engineering and Applications*, 2003, **39**(30): 82 - 84. (in Chinese)

[9] Soo Von-Wun, Lee Chen-Yu, Li Chung-Cheng, et al. Automated semantic annotation and retrieval based on sharable ontology and case-based learning techniques[C]//*Proceedings of the Third ACM/IEEE-CS Joint Conference on Digital Libraries* (*JCDL'*03). Huston, 2003: 61 - 72.

基于语义的构件检索

王 燕 陈 明 赵建辉

(中国石油大学计算机科学与技术系,北京 102249)

摘要:根据构件检索的研究现状,通过增加语义特征,改进了基于刻面分类的构件描述模型. 结合领域本体,提出了基于语义的构件检索过程模型及相应的概念语义匹配算法. 在基于语义的构件检索过程模型中对其中的构件推理引擎、构件分类引擎的实现进行了详细说明,并给出了贝叶斯分类方法在构件分类中的具体应用. 实验表明,基于语义的构件检索方法提高了构件的查全率和查准率,证明了此方法的可行性和有效性.

关键词:领域本体;刻面分类;朴素贝叶斯算法;构件匹配;基于构件的软件开发

中图分类号:TP311