A component model based on sematic for E-commerce PaaS platform

Zhenchao Zhang, Wei He, Qingzhong Li

School of Computer Science and Technology Shandong University
Shandong Provincial Key Laboratory of Software Engineering
Jinan, China
zzc0737@163.com hewei@sdu.edu.cn

***Abstract*—as one of the most important patterns of cloud service, PaaS provides an entire development, deployment and runtime platform for SaaS applications. In a specific PaaS environment such as e-commerce platform, there are similar functionalities among multiple applications, such as "searching items". In this paper, we focus on enhancing general PaaS platform with the mechanism of shared component-based application construction. We discuss some key issues of constructing such abilities with large number of heterogeneous components in the PaaS platform. A component layer with a semantic-enabled multi-level abstract component model is constructed and added to the general PaaS platform. We also give a description of the prototype of our enhanced e-commerce PaaS platform, which can efficiently simplify SaaS application constructing with reusable components.**

Keywords- component, PaaS, ontology,e-commerce

I. INTRODUCTION

As one of the most important patterns of cloud service, PaaS (Platform as a service) provides an entire development, deployment and runtime platform for software providers and their users. ISVs (Independent Software Vendor) deliver and deploy their applications on a PaaS platform, which will be tenanted and customized by tenants. Usually, a PaaS platform based on public clouds supports deployment and running for a large number of applications. For example, in an open e-commerce service platform, there are many different kinds of applications, such as online stores, online payment systems and online logistics. For each of these application types in the PaaS platform, multiple applications can be delivered and deployed by different ISVs and tenanted by e-commerce service providers, such as sellers and logistics enterprises. Generally speaking, there are always some modules with similar or common functionalities among these applications. For example, almost all the online store applications have the functionality of "query product". If some mechanisms for reusing and customizing such common components are provided by PaaS platform, then the ISVs can construct the functionalities by directly integrating available components into their applications without writing their own codes. We believe this ability of PaaS platform can effectively improve the efficiency of application construction for ISVs. To the limitation of our knowledge, almost all of current PaaS platforms haven't provided similar mechanisms for reusing or sharing components at software construction level. In this paper, we focus on some key issues of constructing abilities for reusing components in an open e-commerce PaaS platform.

Component-based software development methodologies have caught the attention of both researchers and developers for a long period of time [1, 5, 8]. Although a lot of online components have been provided in Internet for software developers, it is still difficult for the developers to easily find out the appropriate components according to different informal descriptions of component requirements due to heterogeneous and uncontrollable component specifications and environments. On the other hand, a PaaS platform can provide a relatively unified and standardized environment for component managements and reuse, which makes it possible to research and construct a mechanism for matching, integrating reusable components into ISVs' SaaS applications efficiently.

To enhance the software constructing abilities of e-commerce PaaS platforms with mechanisms based on reusable components, there are inherent challenges. 1. How to smoothly integrate component-based SaaS application development methodologies into general e-commerce PaaS platform. 2. How to effectively manage large numbers of heterogeneous components with informal descriptions and inconsistent semantics, as well as efficiently support application construction with the components. Usually components are developed and delivered with different rules and informal descriptions, so it is difficult and inefficient to find appropriate components to satisfy various needs with a large number of components.

The main contributions of this paper are as follows. 1. We propose an enhanced PaaS platform by adding the mechanism of shared component-based application construction. ISVs can construct their applications with high efficiency and low cost in the platform. 2. An abstract

978-1-4799-3219-1/13 $31.00 © 2013 IEEE
DOI 10.1109/WISA.2013.93

component model with domain ontology is constructed and integrated into the PaaS platform. The semantic component model can be used to describe and manage a large number of heterogeneous components provided by component vendors, as well as provide support for matching components to the requirements of applications.

The remainder of this paper is organized as follows. Section 2 shows the architecture of the PaaS platform with the component layer. Section 3 describes the management and description of the components. Section 4 verifies the realization of the system. Section 5 reviews the related research and section 6summarizes our results.

II. OVERVIEW

Current PaaS platforms provide an entire development, deployment and runtime platform for ISVs. In order to reduce the cost and improve the efficiency for ISVs to develop SaaS application, we propose the mechanism that integrates component delivering and reusing into PaaS platform which can form an enhanced cloud service. We add a component layer into general PaaS platform to support component-based application development for ISVs. Supported by the enhanced PaaS environment, reusable components are delivered by independent providers, which are called ICVs (Independent component Vendor). And we call this SaaS application development pattern CaaS (component as a service), in which ISVs could reuse appreciated components provided by ICVs in their applications via tenanting patterns. Figure 1 shows the overall architecture of the enhanced PaaS platform.

The component layer can be divided into four parts.

- Component library and semantic model. Semantic model specifies descriptive meta-data for business components supported by the platform which is mainly composed by abstract component model and e-commerce ontology. E-commerce ontology describes semantic relationships among terms used in e-commerce domain while abstract component model defines all kinds of components with different functions in e-commerce domain. Components (C) provided by ICVs can be stored in component library with their functional description documents (D), and a component information table (T) is used to store other information of components such as QoS, provider, and interface type.
- Component cluster. COM cluster is responsible for building association relationships between entity component and abstract component which are recorded in association table.
- Component delivery. ICVs can deliver their components with required component information through component delivery interface. And then the Description component describes the component in a unified format.
- Component Binding. Components that can satisfy ISVs' request are selected by the matching engine and the components are ranked by the similarity score. And then ISVs can select a component form these candidate components, the component is selected will be bound to the ISVs' application.

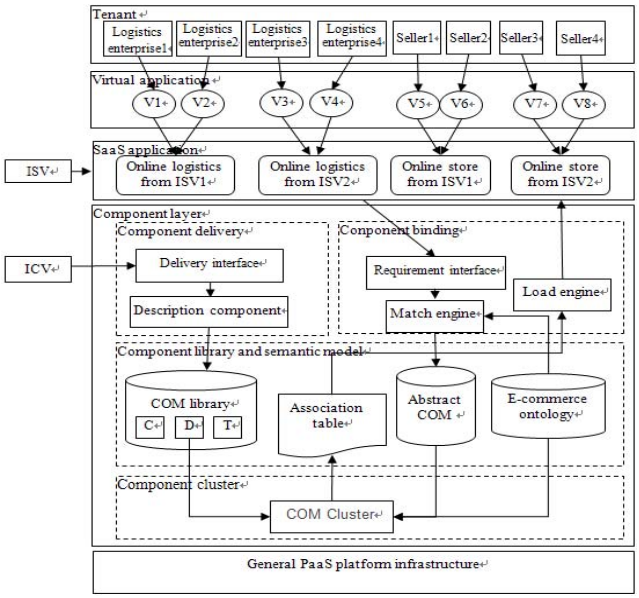

Figure 1. PaaS platform with component development

As shown in figure 1, ICVs delivery components into PaaS platform and ISVs tenant components to construct their own SaaS application. And also ISVs can tenant different components to form different application instance for multi-tenant.

III. MULTI-LAYER COMPONENT MODEL

A. Abstract Component Model and ontology

There are large number of heterogeneous components in the platform which are provided by multiple ICVs with different protocols, interfaces and descriptions. Moreover, there may be multiple business components with similar functionalities and interfaces satisfying the same requirements. In order to manage and describe the components efficiently, abstract component model is constructed. An abstract component is a definition for a particular business component without implementing details, which can be regarded as the agent of a specific cluster of actual business components with similar functionalities.

Definition 1: Abstract Component = (ID, AbInterface, ComponentIfo), the meanings of the elements are:

- ID: the unique identification of the abstract component.
- AbInterface: the interface of the abstract component. The abinterface is described as follow: abInterface=

（Operation, Input, Output）, the meaning of the elements are:

- Operation: operation of the interface. The definition of operation is devided into three parts: action, target, restraint. Action represents the action excuted by the operation, and target is the result of operation wants to get and restraint is the restraints that the operation has to obey. For example, we can define an operation as: action "query", target "product", restraint "in platform".
- Input/Output: the input/output variable set of the abstract component. This set includes all the possible input/output variables of this kind of components. The input/output variables are composed by the essential and optional variable. Each input/output variable is a vector (name, type, optional/essential). Optional and essential are Boolean variable that essential is 1 and optional is 0.

- ComponentIfo: the information about component is mainly including the realization forms of components supported by platform, such as web services, web content, Java Bean, and the if there is an interface of component and the language of component.

For example, we can define the abstract component of "query product" as: abcom= (1, (QueryProduct , ((productname , String, essential),(productID, String, optional)) , ((price, double, essential),(quantity, int, optional))), (web service, null, Java). In order to clearly represent the meaning of the variable, all the variable is named without acronym.

For an e-commerce PaaS platform, there may be many kinds of component. Figure 2 has given out parts of the abstract component model.

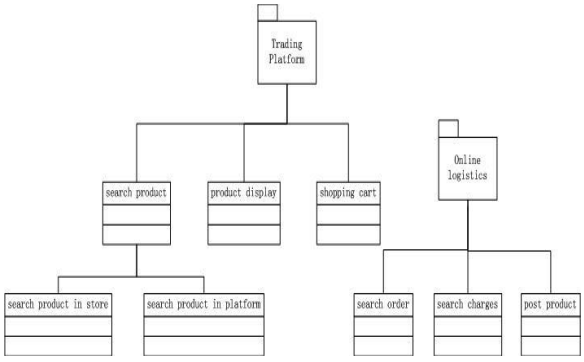

Figure 2. Abstract component model of ecommerce Platform

Although many components implement similar function, it is unlikely that ICVs use a common set of terms to name components and parameters. In order to capture the semantic relationships of terms, we construct commerce ontology. The ontology describes semantic relationships of terms and words. There are two major relationships defined in the domain ontology. First is equivalentclass. This relation includes not only the synonym such as "Product" and "Good" but also some semantic equivalent terms such as "Quantity" and "NumberOfItems" in a special domain. Second is subclassof, it describes the relation between abstract terms and concrete terms such as "Product" and "Clothes". Subclassof is a sequential relation. If A and B are subclassof relation, the similarity of (A, B) is 0.5, but (B, A) is 0.3. Figure 3 has given an excerpt of the domain specific ontology.

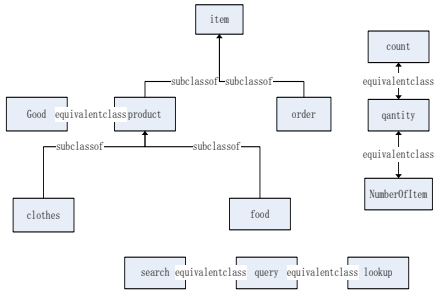

Figure 3. an excerpt of the domain specific ontology

The similarity of the semantic relationship described in figure is shown in table 1,

TABLE I. SIMILARITY OF THE TERMS

term Pair	***Relationship***	similarity
(A,B)	equivalentclass	1.0
(A,B)	subclassof	0.5
(B,A)	subclassof	0.3
(A,B)	Other	0.0

B. Clustering entity components based on semantic model

By computing the similarity between the entity component and abstract component, we can build the association relationship between them. First, we defined the entity component as follow:

Definition 2: Entity Component= (interfacename, input, output, ComponentIfo);

- Interfacename: the name of interface and we usually can get the operation feature of the component by the Interfacename.
- Input /output: the input and output variable of component. For each input/output variable, we have defined optional and essential in abstract model. We will assign a weight to the input/output variable for that essential variable is more important than optional.
- ComponentIfo: the realization form of component. It can only be the forms that the platform supports.

We can get the similarity between concrete component and abstract component by comparing the operation, input

and output, and then cluster the concrete component to the abstract component with the highest similarity score. But developers usually use different coding conventions, so we have to preprocess the terms used to define the information of the component. In order to process the terms, we conclude the methods have been used by many researches:

- Word Segment. Many combined words are used, we have to divided them into single words to understand the meaning of the combined words. According to the rules defined by [12][13], we can divided the combined words into single words. For example, adding space between the uppercase and lowercase, replace "-" with space.
- Acronym Expansion. Sometimes developers use some acronyms for shortness, so we have to expand them to know the full words that developers want to use. In the PaaS platform, ICVs develop components by using the data structure exposed by the platform. So we can construct the mapping between the data construct and acronym.
- Dependent words. Many words may lack of specific meaning, for example, ID, but when it is followed by some other words "order", we can know the specific meaning of it. So we construct a dependent word table to store these words.
- Part-of-speech annotations and filter. According to [], we can annotate each words of terms with the part-of-speech, and then filter meaningless word, for example, preposition"of" and article "a".
- Operation process. There are many definition rules of operation, so we need to format them to unified format. According to semantic model defined in [12], we map operation of concrete component into three parts: action, target, constrain. Usually action is mapped to the verb of the most of the left, target is mapped to a none phrase followed the verb and constraint is followed a preposition.

Given a pair of matching terms (A=SearchProduct, B=QueryItem), the similarity matching score of this pair of terms is calculated based on the following formula:

$$Sim(A,B)=2*match(A,B)/(m+n) \quad (1)$$

Where m and n are the number of valid words in A and B. match(A,B) returns sum of matching score of words that can be matched with e-commerce ontology.

The matching score of entity component and abstract component is used to judge if there is an association relationship between them. The algorithm used to calculate the score is described as follows:

Algorithm 1. similarity matching of concrete component and ahstract component

Input:Q=（operation,input,output）//query component
Output:SC //abstract component that has the highest similarity score
For S_i of S { //S_i is the ith abstract component of S
//operation matching
sim_{op} =1/3*sim(operation.action,Si.operation.action)
+1/3*sim(operation.target,Si.operation.target)
+1/3*sim(operation.restrain,Si.operation.restrain)
//input matching
For $input_i$ of input
sem(i)=max{sim($input_i$,S_i.input)};
$sim_{in}= \sum_i u_i * sem(i)/m$; //$u_i$is weight of $input_i$
//output matching
For $output_j$ of output
sem(j)=max{sim($input_j$,S_j.input)};
$sim_{out}= \sum_j u_j * sem(j)/n$; // $u_j$is weight of $output_j$
//final score
similar(D,S_i)= 1/3*sim_{op}+1/3*sim_{in}+1/3* sim_{out};}
SC={S_i|max（similar（D，S_i））};
Return SC;

Algorithm 1 only consider entity component with a function. If an entity has several functions, it is just the repeat of our algorithm.

We annotate the ID of abstract component that has the highest matching score to the description document of the component, so that we can find the association relation between entity component and abstract component.

IV. COMPONENT SELECTING

In order to improve the efficiency of developing an application, ISVs retrieve reusable components provided by the platform. ISVs submit their requirements when they need a component with a special function. In requirement interface, we divide the needs into three parts: operation, input and output. We have divided the retrieval process into two steps. Firstly, we match the requirement with abstract components to find approriate abstract component. The process of the matching is just like the matching between the entity components and the abstract components. Secondly, with the abstract component we can get many entity components in the association table, and then these components are filtered with the information such as provider, language and interface of components submitted by ISVs, and these components are also needed to be matched with the requirement. At last, we can get a candidate component list that satisty all requirements of ISVs. In order to assist ISVs to select the best component, we rank the candidate components with the score of similarity.

V. REALIZATION AND VERIFICATION OF THE SYSTEM

Figure 4 shows the entire architecture of PaaS platform with component layer. This architecture is based on the PaaS

platform realized by our team in [13]. Our major work is to provide reusable components for ISV in the PaaS platform. As we can see in figure 4, there is a COM subsystem that provides components for ISVs.

In order to get the information of the entity components that delivered by ICVs, we provide an interface (see Figure 5) that ICVs can submit information of their components. ICVs submit component information required with entity component in the interface. The information provided by ICV will be processed as we have proposed in section 3.2, and then stored into the component library.

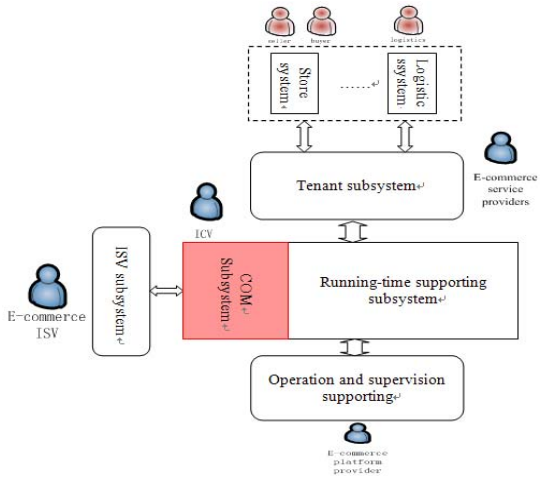

Figure 4. the architecture of the PaaS platform with component layer

As shown in figure 6, we provide an interface for ISVs to submit their requirements. ISVs can submit their various requirements of components such as operation information, interface type and component form. With these requirements, system can return a candidate component list for ISVs.

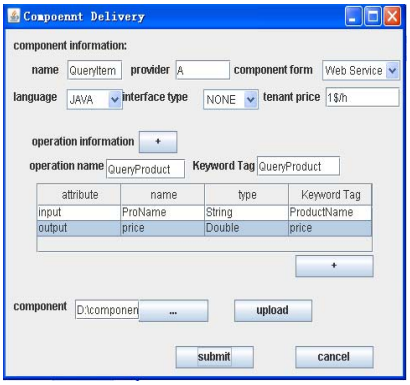

Figure 5. Componentdelivery interface

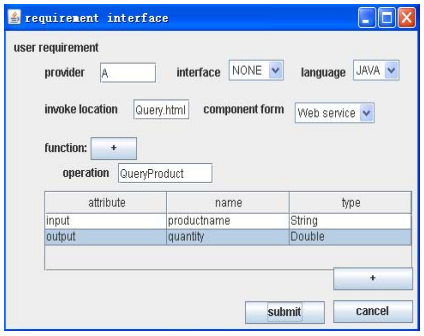

Figure 6. user requirement interface

In order to verify the effectiveness of our approach, we have modeled 500 components with the similar function and retrived these component with different requirement. When we retrive the components without ontology, only 12% of components can get the similarity exceed 80%, and 33% can exceed 60%. But when we retrive the components with ontology, about 35% of the components can get similarity exceed 80%, and 85% can exceed 60%.

VI. RELATED WORK

Component-based software development can not only improve efficiency of development but also realize the reuse of resources. A significant number of research efforts have been devoted to component based software development [1,2,5]. The description and retrival of component are the main questions have to be resolved. One of the description approaches is based on facet. But it is difficult for the manager to find all the facets of a component and it is also difficult for ICVs to describe the component with unified facet standard. But the desciption based on WSDL can be used in PaaS platform for WSDL is a mature standard to be used. As the development of sematic web, more and more researches have used sematic technology to retrive component. With the domain ontology, we can find the sematic relationship between different terms. [2] [3] proposed two matching algorthims. The first uses WordNet to find the relationship between the terms, the second uses the ontology to describe semantic relationships among the terms. Even though WordNet has described the distance about all the words, but some words are domain dependent. So domain ontology is more useful in a specific domain. ICVs may develop component with similar functions, but it is unlikely that they use a common set of name convention. So we have to process the terms used by ICVs. There are many process methods have been proposed, they can be concluded as: word segment, acronym expansion, dependent words, Part-of-speech annotations and filter, operation process. In order to improve the matching degree, we conclude all the method, and we get a better matching degree.

VII. CONCLUSION

In this paper, we focused on the issues of the software development based on reusable component on the open e-commerce PaaS platform. The PaaS platform can better

support various needs of tenants by providing component for ISVs. The main contributions are an abstract component library, matching based on semantic annotation. We classify components by the association relation between components and abstract components. Matching based on semantic annotation can improve the recall ratio and precision ratio. In this paper, we focused on the selection of components but ignored the issue that how to implement data exchange between the components. In the future works, we will do more research on this respect.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61003253, the National Key Technologies R&D Program under Grant No.2012BAH54F04, Shandong Province Independent Innovation Major Special Project under Grant No.2013CXC30201 and Shandong Distinguished Middle-aged and Young Scientist Encouragement and Reward Foundation under Grant No. BS2010DX016.

REFERENCES

1. HEINEMAN George T.,et al. Component-based software development.2001
2. Anne H.H. Ngu, Michael P. CarLson, Quan Z. Sheng, Hye-young Paik. Semantic-based Mashup of Composite Applications. IEEE transactionson services computing, 2010, 2~15.
3. TanveerSyeda-Mahmood, Gauri Shah. Searching Service Repositories by Combining Semantic and Ontological Matching. Proceedings of the IEEE International Conference on Web Services. 2005:13～20 vol. 1.
4. XinPeng, et al. Representing and Retrieving Components based on Ontology. Journal of Nanjing university (natural sciences)，2005，41 (z1)，470~476.
5. Paul C. Clements .From Subroutines to Subsystems: Component-Based Software Development. The American Programmer, vol. 8, no. 11, 1995.
6. Xin Wang. A Review of Web Service Semantics Description Evolution. Modern library and information technology, 2010, 01.
7. Zhendong LI, XianglinFei. Research on the Concept-based Information Retrieval Model. Journal of Nanjing university (natural sciences), 2002, 38(1), 99~109.
8. AbhishekSrivastava, Paul G. Sorenson.Service Selection based on Customer Rating of Quality of Service Attributes.IEEE International Conference on Web Services. 2010,
9. Rouying Zhang, XuesongQiu, LuomingMeng.SLA Representation Method and Application.Journal of BeijingUniversity of Posts and Telecommunications, 2003.
10. JunfengZhao,Yasha Wang, Bin Xie, Fuqing Yang. A Management Framework of Component Supporting QoS of Component. ACTA Electronic Sinica, 2004
11. Lanju Kong, Qingzhong Li. A Metadata-driven Cloud Platform for Delivery of SaaS Application. IEEE International Conference on Information and Automation, 2011
12. Yuanjie Li, Jian Cao. Web Service Classification Based on Automatic Semantic Annotation and Ensemble Learning. IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012.
13. Marco Crasso, Alejandro Zunino, Marcelo Campo. AWSC: An approach to Web service classification based on machine learning techniques. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No 37 (2008), pp. 25-36.