Abstract—In order to adapt to the dynamically changing web service resources in the open environment and ensure the composite service meet the business requirements continuously, the paper presents a service-cluster based approach to service substitution. Substitution means replacing a component with another component, as long as the replacing component produces the same output and satisfies the same requirements as the replaced component. To perform service substitution with less impact on the ongoing, and sometimes critical, business processes, the approach proposes deploying web service cluster. Service cluster contains a logic service and a set of concrete services, these concrete services are functional equivalence or compatibility. The method can quickly replace service; better ensure the normal operation of the combination of services. At last, we make a conclusion and put forward the future work.

Keywords-service-cluster; service substitution; service composition

II. BASIC DEFINITIONS AND CONCEPTS

This section gives some concepts, such as logical service, concrete web service and service cluster. We put forward the formal definition of them as follows:

A. Logic Service

Logic services are abstractions of a set of concrete services. Each service can fulfill certain business functionality. Through hiding the technical details of web services (such as web services’ WSDL address) and encapsulating semantic information, logic services can be regarded as business-level representation of service resources and be understood and operated by business users [4].

Logic service is defined as $ls=\{Bi, Lf,LfNF, refer\}$

- $Bi= (LsID, LsName)$ stands for the basic information of the logic service, including the identifier and the name of the logic service.
- $Lf$ stands for the function of the logic service. $Lf = (funName, input, output, pre, effect)$:
  - $funName$ is the name of the function.
  - $input = \{p1, p2, \ldots , pm\}$ stands for a set of input parameters to Complete function.
  - $output = \{q1, q2, \ldots , qn\}$ stands for a set of output parameters of the function.
  - $pre$ stands for the premise conditional expression for using the function.
  - $effect$ stands for the effect of the function.
A. Concreteservice

In this paper context concreteservice are physical services, we use cs to denote concreteservice, CS denote the set of cs. cs is defined as cs=(Bi, Op,sNonFun, Impl)

- Bi=(concreteserviceid, provider, url, DescriptiveInformation) stands for the basic information of the concreteservice.
- Op = \{ op1, op2, op3, ..., opm \} stands for the operations that the concreteservice provides. opi = (N, IPara, OPara) N is the name of the operation, IPara and OPara are the input parameters and output parameters of the operation.
- sNonFun = \{ snf1, snf2, snf3, ..., snfm \} stands for a set of non-functional attributes of the concreteservice, such as available time, cost and so on.
- Impl stands for the Implementation file of the concreteservice.

B. Service Cluster

Service cluster is to gather web services that address same user’s needs and support their binding through a common interface. A service cluster contains a logical service and a set of concreteservices. These concreteservices are functional equivalence or compatibility. Concreteservices are mapped to logicalservice by service virtualization for example function clustering and transforming. The logicalservice can’t handle business process, which is the logical description of concreteservices. The concreteservice handles business process. Logical services can join service collaboration as concrete service agent.

We define a service cluster with the triple: SC=(LS,CS,R), LS is the logic service, we can indentify logic service with the id of logic service. CS is the set of concreteservices, CS= \{ cs1, cs2, cs3, ..., csm \}. R is the set of correspondences that exist between the logic service and all concreteservices in the service cluster.

The logic service leads the service cluster. A service cluster only has one logic service, and has many concreteservices. The content of service cluster is dynamically changing, new concreteservice may join, and other web services may leave. Some web services may become temporarily unavailable; some web services may resume operation after suspension, etc.

III. SERVICE-CLUSTER BASED METHOD ON SERVICE COMPOSITION

Web services have by default a dynamic nature: they appear and disappear without prior notice and their non-functional properties can change over time. We define substitution as a way for replacing a web service with another, new web service, as long as the replacing new web service produces the same output, and satisfies the same requirement, e.g., non-functional criteria, as the replaced web service. The web service substitution only happens if the following conditions are satisfied: concreteservices are functionally equivalence or compatible and satisfy the function requirement of the logic service. They can be gathered to logic service representative service cluster.

Our web service substitution approach consists of two steps:

The first step is find the failed concrete service depended logic service, and then get the service cluster.

The second step is selecting one concrete service from service cluster to replace the failed services. The selected concrete service must be satisfying the QoS requirement.

A. Relationship of concreteservices and logic services

We give the judge method of concreteservice functionally equivalence or compatible and concrete service satisfy logic service function requirement.

Theorem 1 : functional equivalence marked as: Re \( \in \text{CS.CF} \times \text{CS.CF} \), \( \forall cs_i, cs_j \in \text{CS} \), \( \exists < cs_i.cf, cs_j.cf > \in \text{Re} \), if and only if these conditions are true:

- \( cs_i.cf.D = cs_j.cf.D \), the two concreteservices have the same semantically function description.
- \( \forall I_i = \text{ps}_i.pf.Input, I_j = \text{ps}_j.pf.Input \), \( \exists f : I_i \rightarrow I_j \Rightarrow \) input \( \in I_i, f(\text{input}) \in I_j, f(\text{input}) = \text{input} \)
- \( \forall O_i = \text{ps}_i.pf.Onput, O_j = \text{ps}_j.pf.Onput \), \( \exists g : O_i \rightarrow O_j \Rightarrow \)
output ∈ O_i, g(output) ∈ O_j, g(output) = output

The theorem shows two functionally equivalence services have the semantically function description and them same input and output parameters.

Inference 1.1 \( R_e \) is equivalence relation.

Theorem 2: functional compatibility marked as:

\[
\begin{align*}
\forall c_i, c_j \in CS, &
\exists <c_i.cf, c_j.cf> \in R_e, \text{if and only if the following conditions are true:} \\
&
c_i.cf.D \supseteq c_j.cf.D, \text{ the function of concrete services } c_i; \text{ Semantic include the function of concrete service } c_j; \\
&\forall I_i = c_i.cf.Input, I_j = c_i.cf.Input, \exists f : I_i \rightarrow I_j \Rightarrow input \in I_i, f(input) \in I_j, f(input) \subseteq input. \\
&\forall O_i = c_i.cf.Output, O_j = c_i.cf.Output, \exists f : O_i \rightarrow O_j \Rightarrow Output \in O_i, f(Output) \in O_j, f(Output) \subseteq Output. \\
\end{align*}
\]

in this cases ,the concrete service \( c_j \) can substitute concrete service \( c_i \)

Theorem 3: concrete service and logic service function match can be marked as: \( R_m \subseteq CS.CF \times LS.LF, \forall c_i, l_i \in LS, \exists <c_i.cf, l_i.cf> \in R_m, \text{if and only if these conditions are true:} \\
ls.lf.D \subseteq c_i.cf.D, \text{ the function description of logic service is equals to or the subset of the function description of concrete service } c_i; \\
I_i = l_i.cf.Input, I_j = c_i.cf.Input, \forall input_i \in I_i, \exists input_j \in I_j, input_i \text{ and input}_j \text{ are equal in semantics or } input_i \text{ is the subset of } input_j. \\
O_i = l_i.cf.Output, O_j = c_i.cf.Output, \forall output_i \in O_i, \exists output_j \in O_j, output_i \text{ and output}_j \text{ are equal in semantics or } output_i \text{ is the subset of } output_j.
\]

B. Qos based Services selection

When get the service cluster, the second step is selecting one concrete service from service cluster to replace the failed services. The selected concrete service must be satisfying the Qos requirement. The paper give the judge method.

Algorithm 1:

\[
\text{ValueMatch( )} \\
\{ \\
\text{For every non-function of logic service } \{ \\
\text{flag}=false; \\
\text{for every non-function of csi } \{ \\
\text{if (lnf.name=lnf.name and ValueMatch ( cnf.value, lnf.value)) } \\
\text{flag}=true; \\
\text{if flag=false, return false; } \\
\} \\
\text{return true; } \\
\} \\
\text{Reference [5] presented the common non-functions are price, ExecutionTime,UsefulLife, Authorization, serviceProvider, etc.} \\
\text{With usetime for example, the ValueMatch( ) } \\
\{ \\
\text{if ( Inf.usetime \leq UsefulLife.endTime \text{ and Inf.usetime } \geq UsefulLife.startTime \text{) } } \\
\text{return true; } \\
\text{else } \\
\text{return false; } \\
\}
\]

IV. IMPLEMENTATION

Based on SOA and metadata-driven technologies, we design and implement a service-oriented multi-tenancy platform which is used to deliver economical, easy-to-use supply business management software to small and medium-sized automobile enterprises in the form of SaaS.

Service Component Architecture (SCA) is a programming model for SOA and provides a language-
Service clusters can be replaced with failed services quickly. When a service fails, concrete services in the same logic services are used. Every logic service leads a service cluster. Automobile industry is an example. The SBM consists of the concepts of service substation, we use an application named Supply Business Management (SBM) [12] in automobile industry as an example. The SBM consists of logic services. Every logic service leads a service cluster. When some service fails, concrete service in the same service cluster can be replaced the failed services quickly.

V. RELATED WORK

Web service substitution has received significant attention in the literature. Mecella et al. [6] also introduce a formal model for substitutability of Web services that are represented using state machines. Substitution is considered in the context of a composition where replaceability analysis is done based on trace equivalence [7]. The idea of simulation equivalence is applied by Benatallah et al. [8] for determining compatibility and substitutability of Web services. Martens et al. [9] devise an approach for determining behavioral and syntactical compatibility between Web services that are modeled as petri nets. Here also, the authors adopt trace equivalence and bisimulation equivalence to determine similarity between two petri net models. Taher et al. [10] also determine similarity between two services based on the interface descriptions (specified in WSDL). This work assumes the formation of communities of services that provide similar functionality, and hence can be substituted by analyzing syntactical and semantical similarity between service descriptions. Another interesting approach is reported by Li and Jagadish [11] where the authors adopt graph-matching techniques for substitutability analysis. Their work represents service interfaces using graphs, where nodes and edges correspond to states and transitions of a service respectively, and applies graph-similarity heuristics to determine compatibility between services.

VI. CONCLUSION AND FUTURE WORK

This paper presented our research on web service substitution. We define substitution as a way for replacing a web service with another, new web service, as long as the replacing new web service produces the same output and satisfies the same requirements as the replaced web service. We give the definition of logic service, concrete service and service cluster. The relationship of logic service, concrete service and service cluster are studied in this paper. The judge methods are given. Service cluster gather web services that address same user’s needs and support their binding through a common interface. Concrete services in service cluster can substitute for each other. The method can quickly replace service and better ensure the normal operation of the combination of services.

ACKNOWLEDGMENT

The authors would like to acknowledge the support provided for the project by the National Natural Science Foundation of Shandong Province (ZR2009GM028, ZR2011FQ031), the National High Technology Research and Development Program of China (2009AA043506).

Reference


