Formulating Composition Strategy for Intelligent Web Services

Shen Jianjing         Zhao Huijie         Wang Yuqin         Zhu Weidong
Department of Electrical Information Engineering, Zhengzhou Institute of Information Science and Technology
Zhengzhou (450001), P. R. China
Tel: +86-371-63539251 +86-378-5891155 +86-371-63539276 +86-371-63539267
shenjj@vip.sina.com  zha608@yahoo.com.cn  xwyq_62@sina.com  syzx@163.com

Abstract—For improving the function of Web services, this paper puts forward a method of formulating Web services composition strategy based on Web services components, framework and workflows. It describes the mapping of the composition strategies with requests, subrequests, operations and components, and elaborates the pre- and post-operations of Web services. With the method, we can not only formulate the Web services composition strategy, but also control the composition quality detection of Web services.

I. INTRODUCTION

Intelligent Web services would have the characteristics of self-learning and self-adapting. Over the past few years, although some automatic works at Web services are acquired with simple composite Web services component, they still can hardly satisfy the request of the above two characteristics. So, we need to go further into the study of composition method of Web services components. In common business processing, we usually need to make a plan to arrange the works in a certain period. Based on Web services components, how to establish the Web services that will satisfy the users’ requests is a current problem. D. Beyer and his fellows present a language for specifying web service interfaces, and the interface puts three kinds of constraints on the users of the service [1]; Literature [2] gives a design of verification approach to developing reliable web services, they focus on composite web services which consist of asynchronously communicating peers. The V. Agarwal with his partner have described a two-step methodology for an end to end composition of web services by semantically annotating Web service components, as well as a prototype that demonstrates this methodology in a domain-specific scenario [3]; Literature [4] creates a hybrid genetic algorithm that uses two linear-programming models for evaluation. More and more specialists have made great contributions in this domain. Being a branch of the distributed computing, regardless of peer to peer or grid computing of Web services, if it wants to carry out a high quality system, to make a feasible plan for intelligent Web services is a practical method.

We previously developed models of Web services components based on OWL (Web Ontology Language) and a framework that addressed these combined components [5]. We also developed a workflow management system model for the distribution problem [6]. However, the factors involved in self-learning and self-adapting are often based on experience and feelings, which are difficult to measure and quantify. By using composition plan approach to combine Web services components, we can automatize the composition of Web Services components according to different environment conditions and users’ information, putting the programming reason-

II. USER REQUEST SPECIFICATION

To make a composition plan that matches the user request, we should process overall users’ request with specification method.

There are two subjects needing to be processed about users’ request: request orchestration and specification. A user request usually includes some subrequests. For example, the request of “Video on Demand” (VOD) includes some subrequests like “search the program”, “confirm the contents of VOD” etc. If the user request satisfies the condition of the system, each subrequest with a certain sequence will appear in turn. Generally, this sequence and conditions is implemented by a particularly orchestration model. The description of request is a key factor to automatic composite Web services, which must make its subrequest clear and specific. For instance, the description of the request should understand the meaning of “search the program”.

A. Orchestration Programming Model

We can program the request in two ways: the framework definition programming and system generated programming. In the specification of OWL, the framework is in charge of combining components. It can decide the performance sequence of the subrequests. Similarly with the workflow management system in literature [6], the system generated programming can automatically produce a main engine for composition components. The main engine gives the performance order for each subrequest. The system generated programming is created by composition engine, which gives the performance order to all services to of each subrequest. At this point, we may emphasize that a composition engine is the same as a subrequests when it represents the subrequest orchestration.

Many experts have already put forward many excellent orchestration models in their literatures, such as activity diagrams in UML [7], [8], the grid computing etc. Together with activity diagrams of UML, we program the requests and subrequests. The activity diagram is applied in most models with traditional interactive technology and Web services [9]. From start to end of modeling, all the activities are elucidated by the activity diagrams, which can conveniently facilitate orchestration modeling, and is a better method to process the users’ requests.

The activity diagram means a process of case progress. In our method, each activity represents a subrequest. A subrequest is a component which contains the name of the subrequest. For
example, in the request of “Video on Demand”, user is the sponsor and the end of the affair. This composition request usually includes five subrequests: 1) register; 2) order plan arrangement; 3) electronic invoice; 4) detailed program; 5) program broadcast.

In other ways, this case can also be implemented as client/server. Through “fork” and “join”, the activity diagrams can establish some serial or parallel pipelines. In the activity diagrams, “fork” is a split of a composition component, and is the start position of the next parallel pipelines.

A composition request may be processed as a subrequest. For instance, a user request is to translate the dialect of London into the dialect of Shanghai. Usually, a person who knows English does not necessarily understand the dialect of London, and a lot of Chinese do not understand the dialect of Shanghai. This request includes a series of subrequests from the dialect of London to the dialect of Shanghai. If Web services have the integrity function and can be discovered by the users, subrequests probably map to the particular Web services. The translation from dialect of London to dialect of Shanghai will be mapped to the following Web service operation:

London dialect → English → Chinese → Chinese → Shanghai dialect.

B. Describe the Subrequest

The activity diagrams have the function to describe performance order of the subrequest specification, and also need to provide the description method of each subrequest, such as the description of the syntax and the semantic. We define the subrequest ontology for this subject. All subrequests are described through ontology based on OWL. A subrequest can map to one or several Web services components. After describing subrequests by ontology, each subrequest becomes the ontology that includes the syntax, static semantic, dynamic semantic and quality. According to the related theorems, algorithm of composition request description based on OWL described as follows.

```xml
<?xml version="1.0" encoding="GBK"?>
.rdf:RDF xmlns="http://a.com/ontology#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl=http://www.w3.org/2002/07/owl#
xml:base="http://me.com/ontology/request">
<request>
  <name>......</name>
  <description>......</description>
  <parameters>......</parameters>
  <subrequest>
    <syntax>......</syntax>
    <staticSemantics>......</staticSemantics>
    <dynamicSemantics>......</dynamicSemantics>
    <quality>......</quality>
  </subrequest>
  .......
</request>
</rdf:RDF>
```

This algorithm is a general form of a user request. With the specification of OWL, it describes users’ request and subrequests by ontology. Then it produces an XML file. In the XML file, the user request (<request>) includes the name of request (<name>), request description (<description>), parameters (<parameters>) and the subrequest (<subrequest>). The main factors of the subrequest are as fellows: (syntax, static-semantics, dynamic-semantics, quantity, input, output). The syntax element includes the attribution, the limit of subrequest bind etc. The static-semantics and dynamic-semantics are used to describe the static and dynamic semantics of the subrequests. Because a subrequest inherits the environment condition of the request, the dynamic semantics of the subrequest must accept the restriction of the requests. The quantity element includes the attribution of the subrequests. The input and output elements describe the essential input and output parameters of the subrequests.

C. Profile and Web Services

Client self-determination is a main function in intelligent Web services. Suiting with the characteristic of the user’ request is necessary for composition components. The characteristic is based on profile:

Profile {Coordinates Value <LC>, Rule Matrix<RM>, Degree value<degree_num>, Bind Value<bind_num>, Plan Value<max-plans>}

The “Rule Matrix” provides the components binding for Web services. As a complement of the profile, the “Rule Matrix” works as a framework which assigns the position (or coordinates) for a component. Moreover, the framework also needs to provide a series of input parameters to be used in the performance period of the composition Web service (for example, the program name of VOD, the place position of VOD server, the distance between user and server etc.). These parameters are explained by its name, data type, unit, role and languages.

III. AN INSTANCE-BASED FRAMEWORK FOR COMPOSING WEB SERVICES

As we have previously mentioned, framework technique is one of the mature techniques for components management. However, the ontology among the Web Services component may take more effect, such as setting the right position of the Web Services components in framework, designing the specification service at a knowledge level and constituting the Semantic Web Service.

A pay-TV user wants watch the famous film “Gongfu”. He chooses and plans the time through the Internet by himself. There are four components: “register & order”, “electronics invoice”, “make order” and “watch & order plan” in VOD management system. The processes of VOD system are that the output information “program contents” of the composite component “watch & order plan” is constructed by combining the output information “order” of the component “make order” with the output information “electronics invoice” of the comp-
ponent “register and order”. The input information of the composite component “watch & order plan” can be divided into two parts: one is the input information of “make order” and the other is “user message” of component “register and order”.

\[
\begin{array}{cccccc}
& \text{Verify register} & \text{Verify credit cards} & \text{Select film} & \text{Select timetable} & \text{Make order} & \text{Watch & order plan} \\
(2) & & & & \text{Web services component warehouse} & & \\
& \text{Axioms} & \text{Instances} & \text{Taxonomy} & \text{Instances model} & \text{Register and order} & \text{E-invoice} \\
& \text{Translation model} & & & & \text{user} & \\
& & & & \text{Film-name & play-time} & & \\
\end{array}
\]

Fig 1. FWSC and a VOD service

A. A Framework for Composing Web Services

It’s easy to set the right position about VOD instance and the relation of these components by hand. However, the computer cannot understand the meaning of the components. Ontology among the component may describe the component clearly [5]. Therefore, we’ve designed a framework to help users filter and select services while building the semantic Web services (see figure 1).

In the warehouse, the Web service components are constructed by the atom components and their ontology descriptions are made by OWL and RDF. When a user’s requirement appears, according to the semantic requirement, this requirement is disassembled in instances and combined ontologies by the Instance Model. These instances are submitted to Check Model for verifying whether these instances are suitable for the axioms. If they obey the axioms, we can use the specification of OWL and RDF to translate these instances. After that, by selecting the appropriate components from the Web service components warehouse, we can create the composite Web services that will matched the user’s requirement.

B. The Interfaces Bind

Another key problem of Web service component composition is the interfaces bind for the compositions of the atom components and the ontologies denotation.

With the spread of the methods presented in [5], [6], we extend the methods to the interfaces binding with more atom components, and discuss the problems of reducing the number of interfaces and types.

C. Ontologies Composition

After the components have been combined, the ontology information of components will be change with the new composite components. The key point is how to use the ontologies in each single component to construct a new ontology that serves the new composite Web service. The characteristics of ontology are a formal specification of a shared conceptualization. Ontologies of composite Web service cannot be simply added to an ontology of a certain component, it has its own rules that are as follows:

- The information synthesis: Filter the combined ontology elements to output information, in order to format information in composite Web services.
- The information decomposition: Construct input message of composite Web service, in order to produce the input elements of Web service.
- The information mapping: We could specify a mapping between input elements and output elements of Web services.

D. FWSC Operation

Let’s suppose in the instance we have previously mentioned, through Semantic Web Services environment, a user designs a service called VOD. The goal is to watch the famous film “Gongfu” (film-name) at 22:00 this evening (play-time). Figure 1 shows this service’s process using the FWSC environment. The user inputs the film name and the time he wants. The machine outputs the verifying information and electronics invoice. The VOD service comprises four atom components: select film, select timetable, and verify register and credit cards information. If he is a normal user and have enough money, the action of VOD service will be successful.

When a user inputs the film name “Gongfu”, among FWSC, Instance Model, Checking Model and Translation Model begin to analyze the domain of this requirement. After various processes as described above, FWSC will be divided into eight appropriate Web services components from among the Web ser-
services components to execute this task (see figure 1).

IV. THE MATCH OF REQUEST, COMPOSITION PLAN AND WEB SERVICES COMPONENTS

We previously have tried to directly mapping users’ requests to Web services components. After simply processing users’ requests, we check Web services components according to semantics of ontology. Although it is easy to combine Web services, it is hard to improve the service function. With little record text of service, the upgrade of Web services looks difficult. Particularly, there are no necessary design for the self-learning and self-adapting of the Web services, which helps us to realize the importance of composition plan.

From the profile of the user request, we can extract main factors of the user request. The extractive operation can split the user requests into series subrequests. At this moment, each subrequest should map a composition plan.

With profile of user and composition plans of subrequests, composition plan can be made by the following algorithm:

\[
\text{For (each subrequest } \text{SR}) \text{ do} \{
\]

1. Subrequest match (Profile) {
2. For (each subrequest SR) do {
3. 1:1 match composition; (SRi, Plans 1:1, Profile, num plans)
4. } 40 If num plans ≤ Profile.max plans
5. Then 1: N match composition; (SRi, Plans1: N, Profile, the num plans);
6. } 60 For each subrequest, the algorithm tries to match it with a composition plan in a 1:1 composition. At this time, each composition plan and subrequest is combined by a 1:1 plan. If the number of the max_plan is less than or equal to the plan number of these subrequests, it means the composition plans can not satisfy all users’ request of subrequest, the algorithm may match a plan with one to more (1:N) mapping.

A. One to One Match Composition

According to the characteristics and information of subrequest, we have three ways to take matching algorithm of 1:1. The first way is for operation match, which compares all subrequests with all operations. The second way of the match is for operation match, which compares all subrequests with all operations. The second way is for operation match, which compares all subrequests with all operations. The third way adopts the method of composition information. After comparing (D_i) information and subrequests which match the information, it determines which (D_i) can take part in an input operation. Because of the space limit, we only introduce the thinking of the algorithm.

In registered Web services components of a database, each component can complete a certain operation function. So, the operations and Web services components are 1:1 (or 1: N, N: 1). Composition matching algorithm based on operation that takes operation as the foundation, building the relation between subrequests and operations. We can also extend to subrequest and Web services components. The algorithm with exhaustive search to check all operations that match the subrequests, and then compare the request of this operation can satisfy the subrequest all operations that can replace the subrequests will be decided.

In the processing of Web services construction, we realize that satisfying the user request is the first step for the intelligent work. We still need to consider the recessive user request, to improve the Web services in logic rule and axiom. Not only has the function of user been mentioned, but also the pre and post operations have to complete Web services.

For some special user requests, all of their activities are limited in a special domain. Domain ontology may help match algorithm with request, and the establishment of the request ontology contributes to the realization of this algorithm. It only browses the relevant domain to deduce the number of operation, and decide all input operations that can replace the subrequests.

Because all operations of the request are appropriate operations in a certain domain, as a keyword, (D_i) is a criterion in this match algorithm. All relevant search operations are limited in the domain. Then we can decide all input operations (op_i). The general operation can replace the input operation. All of these input operations are actual operations that are provided by the domain. Moreover, the input operation description is different from that of a match operation. It deletes the input/output parameter. The syntax, semantics, information, function matches is similar to the operation-based algorithm.

The match algorithm based on the information is clearer than the above two algorithms. It uses composition information to replace the operation, and then gets the operations that can match subrequests.

At first, match algorithm based on information will resolve two important problems: (1) dominate information modeling; (2) deduce operations from corresponding information. A finite state automaton could clearly describe this subject. Usually, a finite state automaton is a five-tuples (Q-state, Σ-alphabet, δ-transition function, q₀-start state, F-set of accept state).

Let’s design a finite automaton FA, using subrequest, XML and domain member (D_i). The data type of XML is alphabet. We may define a Deterministic Finite Automaton (DFA) FA, for each (D_i). FA creates the model for input/output information of domain member (D_i). If we input a (D_i), FA will output an (op_i).

B. One to More Match Composition

In most situations, one-to-more match composition has wide application scope. Diverse subrequests bring a different operation set. As we have previously mentioned, the translation from dialect of London to dialect of Shanghai usually includes three operations: (1) London dialect to English; (2) English to Chinese; (3) Chinese to Shanghai dialect.

For 1:N match composition, we can link all operations through a flow chart. The node of the flow chart has two types: Operation and mix node (see figure 2). The arrow in this flow chart only connects different types of the nodes. Operation node only connects to hybrid node and vice versa. In our chart, no operation node connects to other operation node, and no hybrid node connects other hybrid node.
At the end of the flow chart, we can decide which operation nodes connect with hybrid nodes And1, And2. The parameters of And1, And2, English and Chinese will be described by ontology. Three processing (London Dialect → English, English → Chinese, Chinese → Shanghai dialect) will relate three operations. The first operation translates London dialect to English, the second operation translates English to Chinese; the translation from Chinese to Shanghai dialect is the third operation.

![Language translation flow](image)

As we see in figure 2, the hybrid node And1 is connected to op1 and op2, the And2 connected the op3 and op4. If we validate this chain, all work can immediately be completed by the operations that are specified by the chain. At this time, composition plan can create a path that maps request and operation, namely: London Dialect → English → English → Chinese → Shanghai dialect.

V. COMPOSITION PLAN DESCRIPTION

After the matching process, each subrequest will generate one or several composition plans. The target of this stage is to use composition plans to replace the subrequests. At first, we select a composition plan for each subrequest. Then it generates Web services description according to these selected plans. Generated processing contain three stages: first, it selects composition plan for each subrequest, then inserts pre and post operations for these composition plans. Third, we measure parameter of composition plan to control the quality of Web services. If we do not satisfy composition plan, the FWSC management system can choose another plan to generate a new plan with Web services description.

A. Composition Plan Replacing Subrequest

Suppose a composition plan has already been selected by a subrequest. Making use of these activity diagrams and plan, we can build an activity diagrams for composition Web services. Each subrequest can be replaced by a proper operation in the plan. If this plan is acquired through 1:N match composition, the performance sequence of these operations is just like the plan and the flow chart.

B. Pre and Pre-Operation

Assume that all subrequests are replaced by their plans in the activity diagrams, the next step to generate Web services is to inserted the pre and pre operation into a composition plan. As a constant pre-operation, self-learning and self-adapting may be inserted in the plans. Because the characteristics of self-learning and self-adapting are varied according to the environment, it appears new characteristics are continuously in process. So, composition plans of the related components also need to be continuously perfected and renewed, which decides the pre and pre operation of self-learning and self-adapting.

Assume (opj) is an inserted operation, we may find (opj) in operation set. If (opj) is found, we can decide all operations that have relation with pre and pre operation. In order to attain this purpose, we also need to select all operations that include relevant path sub-diagrams. Dr. Floyd-Warshall provides a dynamic procedure which can get the sub-diagram [10]. In this sub-diagram, we still need to establish the activity diagrams and transition function.

C. Composition Quality Control

When activity diagrams of composition Web services plan have been made, our next work is how to inspect the quality of the Web services. With the model of FWSC composition framework, we define four main parameters to inspect the quality of composition: The robustness of composition, ranking of compositions, completeness and intelligence of composition.

The robustness check—to inspect work ability of composition Web services components in the different environment. The FWSC is a basic platform of generating composition Web services. In the FWSC, the Check Model is a specialized component to control quality. With this component, we can acquire the robustness of the composition Web services.

To inspect the ranking of Web services components, we can again take a look at the Instance model in the FWSC, in which the workflow management also provides a better method to guarantee the ranking. The sequence of tasks in workflow is well established, and the ontology has an explicit description. The performance sequence of the Web services is certain and careful.

The integrity of Web services is one of the most important parameters. Making composition plan is a perfect way for this requirement. Subrequests can map to composition plans, and composition plans can map to Web services components. The Web services become the embodiment of all user requests.

Concerning intelligence, it is continuously perfect process. The component based on OWL has already provided the interfaces and method of ontology with a study function. By accumulation of ontology, the function of self-adapting would automatically increase.

VI. RELATED WORKS

The Web service component has a simple interface and ontology information and it is combined by atom component, ontology and RDF / OWL [5]. The RDF is used to annotate the component and ontology information for describing the component function and the data model of relationship between components, and providing the simple semantic description for the data model. These data model interfaces are expressed with the OWL, and then get the Web service component based on
the user requirement, decision supports, user relation and e-business.

IBM’s WebSphere, Microsoft’s .Net, BEA’s WebLogic, Sun ONE, HP’s NetAction, Oracle9I AS as the middleware or development environment are under way to create support for composition of Web Services. They extend interaction models to define a process that provides and consumes multiple Web Service interfaces [11]. Evren Sirin, Bijan Parsia and Jim Hendler describe an interactive composition technique to filter and select semantic Web Services which assisted composition approach to use the richness of semantic Web service description and information from the compositional context to filter matching services and help select appropriate services[12]. The OWL-S augments current UDDI architecture with semantic service descriptions[13]. Jim Blythe, Ewa Deelman and Yolanda Gil provide a planning system to select application components and computing resources to help generate executable workflows for a grid, and provide different levels of support, depending on the information available [14]. Jihie Kim, Yolanda Gil, and Marc Spraragen present an interactive workflow composition tool named CAT. This tool combines knowledge-based representations of components, together with planning techniques that can track the relations and constraints among components [15].

Srini Narayanan and Sheila McIlraith define the semantics for a relevant subset of DAML-S in terms of a first-order logic language, obtaining a set of axioms that describe the service features [16]. The Web service components are created under the Internet protocol. So it does not depend on the development environment, hardware and operating system. The Web services will have the characteristics of opening, being commonly used and loosely coupled, achieving by combining the components. With these features, it will provide a convenient way to establish Web services application systems. Asunción Gómez-Pérez and Rafael González-Cabero[17] present a solution using a framework that enables Semantic Web Services design and composition at the knowledge level in a language independent manner. Microsoft’s Global XML Web services Architecture (GXA) has debuted in a new SDK. The company has made it available to developers. GXA is a set of technology aiming to make Web services appropriate for application integration across platforms and over the Internet [18].

VII. CONCLUSION

Chinese proverb says: “Beforehand preparation leads to success, unpreparness results in failure.” Making composition plan is an important content in our intelligent Web services study. Although it has quickly developed in semantics Web and ontology, the intelligent Web services still have many difficulties to overcome. To establish a careful composition plan for Web services all requests of the user, axiom, habit and so on should be considered. The study of intelligent Web services should be quickened.

ACKNOWLEDGMENT

This work is supported in part by NSF of Henan, P. R. China.

REFERENCES