Journal of Southeast University (English Edition) Vol. 22, No. 3, pp. 306 – 310 Sept. 2006 ISSN 1003—7985

# Problem of interoperability in semantic web service system

Man Junfeng[1,2] Peng Sancheng[1,2] Xiang Jianwei[1] Hu Yongxiang[1]

([1] Department of Computer Science, Hunan University of Technology, Zhuzhou 412008, China)

([2] College of Information Science and Technology, Central South University, Changsha 410083, China)

**Abstract:** In order to fully realize semantic interoperability among distributed and heterogeneous applications on the web, a set of effective interoperability mechanisms is presented. This mechanism adopts service interactive interfaces (SII) and service aggregative interfaces (SAI) modeled with abstract state machine (ASM) to abstractly describe the behavior of the invoked web service instances, which makes business processing accurately specify tasks and effectively solves the problems of communication and collaboration between service providers and service requesters. The mechanism also uses appropriate mediators to solve the problems of information and communication incompatibility during the course of service interaction, which is convenient for service interoperability, sharing and integration. The mechanism's working principle and interoperability implementation are illustrated by a use case in detail.

**Key words:** semantic services-oriented architecture; abstract state machine; service interface; interoperability

The semantic service-oriented architecture (SSOA) fusing semantic web and web services is a new-style software architecture. It is convenient for computers either to understand services or implement high level operations, e. g. , intelligent reasoning. In order to really achieve this goal, the SSOA needs advanced semantic interoperability, however, the current interactive model of interoperability is far from satisfactory.

At present, the typically interactive model includes BPEL4WS[1] and WS-CDL[2]. The former distinguishes between two ways of describing the business processes: executable business processes and abstract business processes. These processes use a messages mechanism to exchange the behaviors of a partner role in an interaction. The latter defines a non-executable message exchange between partners in web services collaboration. The main defect of the above two models is the failure to explicitly specify the internal behaviors of partners. It is difficult for the system to accurately designate tasks. At the same time, it obstructs the implementation of services composition and the reuse of business processing pattern. Their capacities for interoperability are no doubt limited.

In semantic web services (SWS)[3], service interfaces are understood as decompositions of a service capability that describes how the service functionality can be consumed and how it is achieved. W3C firstly presents the notions of choreography and orchestration[4] that are regarded as a common basis for describing the dynamics of web services usage and web services interactions, respectively. The choreography part concerns the interactions of services with their users. Any user of a web service, automated or otherwise, is a client of that service. For the convenience of illustration, we entitle this type of interface as a service interactive interface (SII). The orchestration part defines the sequences and conditions in which one web service invokes other web services to implement some useful function. That is, an orchestration is the pattern of interactions that a web service must follow to achieve its goal. For the convenience of illustration, we entitle this type of interface as a service aggregative interface (SAI). SII and SAI effectively overcome technological defects of previous models, and they conform to the requirement of complex business processes in SWS.

In the SSOA, the key problem of SWS interface is how to accurately describe interfaces and use SWS technology to automatically process them. On the basis of the concepts of SII and SAI, we adopt a state-based mechanism—abstract state machine (ASM)[5] to abstractly describe the internal behavior of the service with respect to an invocation instance of a service. We have chosen an abstract machine model for the description of this interface since such a service invocation may consist of a number of interaction steps where these possible interactions can be abstractly described

**Received** 2006-04-25.

**Foundation items:** The Natural Science Foundation of Hunan Province (No. 05JJ30122), the Education Department Foundation of Hunan Province (No. 05C519).

**Biography:** Man Junfeng (1976—), male, graduate, lecturer, mjfok@tom. com.

by a abstract state machine. We also need to set up a strict measures to control and manage the initialization and state transition of ASM. By controlling and managing the ASM model, we can easily implement the automatic processing of the interface. Furthermore, the SSOA model emphasizes semantic mediation and decoupling of web services, so we should design mediators with comprehensive functionalities to resolve heterogeneous problems which naturally result from an open environment.

## 1 ASM Model

ASM, formerly known as evolving algebras[6], provides a means to describe the system in a precise manner using a semantically well founded mathematical notation. The core principles are the definition of ground models and the design of systems by refinements. Ground models define the requirements and operations of the system expressed in a mathematical form. Refinements allow for the expression of the classical divide-and-conquer methodology for system design in a precise notation which can be used for abstraction, validation and verification of the system at a given stage in the development process.

The ASM-based model provides the formal basis for specifying ontology data interchange within service interfaces. In accordance with the requirements of the service interface, the model consists of three notions that we formalize in the following definitions:

① A vocabulary $\Omega$ that defines the information space of a service interface on the basis of the ontologies. It is defined as an ontological schema of the information interchanged in a service interface by denoting the used concepts, relations, and functions of the ontologies. The communicative usage of this ontological schema information is indicated by sub-information spaces for ontologies instances: $\Omega_{\text{in}}$ denotes the information received by the service interface, $\Omega_{\text{out}}$ denotes the information that is provided by the service interface, $\Omega_{\text{shared}}$ denotes the information both received and provided by the service interface, $\Omega_{\text{static}}$ denotes the ontology notions that cannot be changed by the service interface, and $\Omega_{\text{controlled}}$ denotes those that can only be changed by the service.

② States $\omega(\Omega)$ that denote a status of the information space within the dynamics of a service interface that is defined by the attribute values of the ontology instances of $\Omega$. A state denotes a stable status within the dynamics of a service interface that is existent as long as the attribute values of instances are not changed.

③ Transitions rules (TR) that specify the dynamics of a service interface. The general structure of TR is: IF the condition is satisfied THEN the state will be updated, the condition reflects changes in $\Omega$, while the update part defines the changes to the information performed in the transition to the subsequent state $\omega'$. At a state change from $\omega$ to $\omega'$, all TR are executed when the condition is satisfied.

We use ASM to model the services interface of SWS, on the basis of which, we can easily implement interaction and collaboration of services. The obvious advantages of the method are as follows: ① To provide representation means for the dynamics of the information interchange that takes place when a service is used and interacts with other services; ② To support ontologies as the underlying data models along with an appropriate communication technology for information interchange; and ③ To rely on a sound formal model that defines the semantics of service interfaces to allow operations on them.

## 2 SII Model

We have addressed that the SII deals with interactions of the web service from the client's perspective. We use ASM to describe the behaviors of the web service.

### 2.1 Basic description of SII

We denote a single service interface as $\text{ci}(S)$. From a global perspective, the communicative interaction process of several web services and clients via their respective $\text{ci}(S)$ is entitled service processing which is denoted as

$$C((S_1), \ldots, (S_n)) = \text{interaction}(\text{ci}(S_1), \ldots, \text{ci}(S_n))$$

Convenient for web services processes and interaction with the client, we should present external visible behavior, protocol and semantic representation which are required in the interactive process. We define $\Omega_{\text{ci}(S)}$ as the ontological information space of the SII, and the transitions rules $\text{TR}_{\text{ci}(S)}$ for defining the communication protocol expected for client service interactions along with the ontological information interchanged. The external visible behavior is implicitly defined by the states $\omega_0(\Omega_{\text{ci}(S)}) - \omega_n(\Omega_{\text{ci}(S)})$ that can be reached by respective $\text{TR}_{\text{ci}(S)}$. Thus, we can define the interaction of service interface from a global perspective: $\Omega_{C((S_1), \ldots, (S_n))}$ defines the ontological information space used for interaction, $\text{TR}_{C((S_1), \ldots, (S_n))}$ defines the communication protocol and semantics of the information interchanged, and the states of the business process

are defined implicitly as above.

### 2.2 Compatibility of SII

In order to truly implement interoperability, we should consider the problem of capability. This means that given $\mathrm{ci}(S_1), \ldots, \mathrm{ci}(S_n)$ for web services $S_1, S_2, \ldots, S_n$ that shall interoperate, each $\mathrm{ci}(S)$ is considered as static (i. e. not changeable with respect to the web services functionality). We have to determine whether there exists a global SII which regards $C((S_1), \ldots, (S_n)) = \mathrm{interaction}(\mathrm{ci}(S_1), \ldots, \mathrm{ci}(S_n))$ as a valid interaction protocol for the web services. Convenient to discuss SII, we denote service compatibility (SC) as $\mathrm{SC}((S_1), \ldots, (S_n))$.

To determine this, we need to prove that there exists a $C((S_1), \ldots, (S_n))$ for which the following holds: ① The service interface descriptions need to use homogeneous ontologies; ② The information interchanged (i. e. the content of communicative acts) needs to be compatible. We refer to this as information compatibility, denoted as $\mathrm{SC}_{\mathrm{info}}(C)$; and ③ The communication protocol of the SII has to be sound, meaning that it has at least one start state and can reach a termination state without any additional input. We refer to this as communication compatibility, denoted as $\mathrm{SC}_{\mathrm{comm}}(C)$.

We can determine information compatibility on the basis of vocabulary definitions in the SII. Considering this for two web services, information compatibility is obtained if all information required as input by $S_1$ is output of $S_2$ and vice versa. The following gives the formal definition for information compatibility:

$$SC_{\mathrm{info}}((S_1),(S_2)) \leftarrow (S_1(\Omega_{\mathrm{in}}) = S_2(\Omega_{\mathrm{out}})) \vee (S_1(\Omega_{\mathrm{out}}) = S_2(\Omega_{\mathrm{in}})) \quad (1)$$

$$SC_{\mathrm{info}}(C) \leftarrow \forall S_x \in C,\ \exists S_y \in C(SC_{\mathrm{info}}((S_x),(S_y))) \quad (2)$$

For communication compatibility, we should firstly consider the following problems: ① There is an initial state $\Phi$ for $C((S_1), (S_2))$ that is compatible with $\mathrm{ci}(S_1)$ and $\mathrm{ci}(S_2)$; ② The second one is compatibility of TR in each state $\omega_x(C((S_1), (S_2)))$. In a state $\omega_x(\mathrm{ci}(S_1))$, given IF-part of all TR, there exists a TR whose THEN-part fulfils equivalently subsequent state, $\omega_x(\mathrm{ci}(S_2))$, or vice versa; and ③ For $C((S_1), (S_2))$ should exist a correct termination state, i. e., there is no such TR which makes $\omega_x(C((S_1), (S_2)))$ transform into $\omega_x(\mathrm{ci}(S_1))$ or $\omega_x(\mathrm{ci}(S_2))$. We define communication compatibility as follows:

$$\mathrm{SC}_{\mathrm{comm}}(C) \leftarrow \forall S_x \in C,\ \exists S_y \in C(\phi(S_x, S_y) \wedge \mathrm{TR}_{w_1 - w_m}(S_x, S_y) \wedge w_t(S_x, S_y)) \quad (3)$$

$$\phi(S_1, S_2) \leftarrow \phi(S_1) = \phi(S_2) \wedge \mathrm{TR}_{\phi}((S_1), (S_2)) \quad (4)$$

$$\mathrm{TR}_w(S_1, S_2) \leftarrow a \in \mathrm{If}(T_w(S_1)) \wedge b \in \mathrm{Then}(T_w(S_2)) \wedge SC_{\mathrm{info}}(a, b) \quad (5)$$

$$w_t(S_1, S_2) \leftarrow \neg\ \exists \mathrm{TR}(w_t) \wedge w_t(S_1) = w_t(S_2) \quad (6)$$

On this basis, the service compatibility is determined with information compatibility and communication compatibility.

### 2.3 Mediating the Compatibility

In the SSOA, there are two types of mediators: ① Data mediator (DM): it mediates data heterogeneity that may occur in service interaction. It is an ontology-to-ontology mediator (ooMediator). ② Processing mediator (PM): it analyzes two consuming instances at running time, and mediates possible mismatching. It includes goal-to-goal mediator (ggMediator), webService-to-goal mediator (wgMediator), webService-to-webService mediator (wwMediator). In the course of service interaction, the central functionalities of mediators include: ① Providing a homogeneous information space for an SII by solving terminological mismatches between the interfaces of participating web services; ② Handling missing information with regard to information compatibility; ③ Establishing a communication compatibility if this is not given in advance. Processing incompatibility with mediators is shown in Fig. 1.

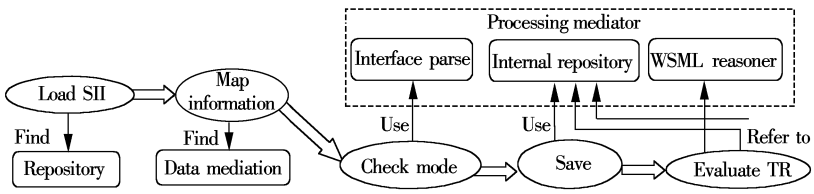


**Fig. 1** Using PM to process compatibility

The PM includes three parts: interface parser, WSML[7] reasoner and internal repository.

The SII parser has the role of determining if any instance obtained after the data mediation process is expected by the targeted partner. It performs the following operations: ① Determining the owner of a certain instance, and ② Determining the value of the mode attribute for the owner. If the value is set to in or shared, the instance will be stored in the internal repository for further usage.

The internal repository is used for storing information that will be sent to one of the partners at some

point in time. It offers the methods of storing, retrieving, deleting and updating information.

The WSML reasoner is the most complex component of the PM. Its task is to extract one by one the instances from the repository, and to check if by sending that instance at least one condition of one transition rule can be fulfilled.

## 3 SAI Model

It is envisioned that the SAI should make use of the ASM model to describe the interactions between web services and goals. In order to link to externally called services or (sub) goals that the service needs to invoke to fulfill its capability, we need to extend the SII as follows: ① Respective goals or services are executed in parallel to other rules in SAI; ② The state signature defined in the SII can be reused; ③ The state signature for SAI can extend the state signature of SII with additional in/out/shared/controlled concepts which need to be tied to the used services and rules by mediators; ④ Respective wwMediator or wgMediator need to be in place to map the in and out concepts defined in SAI to the respective out and in concepts of the SII in the used services and service templates.

The SAI defines a decomposition of the capability of web services and how these subtasks can be achieved by using other web services. Thereby, only those subtasks of a web service functionality are defined in SAI($S$) that are realized by aggregating other web services.

An SAI($S$) = aggregation(($S_1$), …, ($S_n$)) defines the control and data flow for aggregating web services ($S_1$), …, ($S_n$) so that the functionality of $S$ is achieved; thereby, $S$ consumes the aggregated web services via their respective SII ci($S_1$), …, ci($S_n$). We assume that a specific functionality required in a state $\omega_{SAI}(S)$ may only be a part of the functionality provided by an aggregated service. Hence, we introduce the concept of an operation for describing the interaction between services for consuming partial functionalities within an SAI. An operation op(service, update) denotes the interacting services and the communicative activities performed. Thus, an SAI description consists of the vocabulary $\Omega_{SAI}(S)$ that denotes the information space of the SAI from the perspective of the orchestrating web service $S$, transition rules $TR_{SAI}(S)$ of the form "If condition ($\omega_{SAI}(S)$) then op(starget, update)", and the states $\omega_x(SAI(S))$ that can be reached by $TR_{SAI}(S)$.

## 4 A Use Case

We have illustrated SII, SAI and appropriate mediation measures which provide a comprehensive interoperability mechanism. In order to testify its feasibility and practicability, we design a use case of the SSOA model, i.e., online healthy consultation system (OHCS). The system is a typical usage scenario that provides the consulter (Co) with access to the advice services offered by specialists organized in the community of services (CoS). This access must be enabled on demand and it must allow the Co to select a CoS to arrange the meeting. The selection criteria depends: ① On the correspondence between the problem the Co is looking for advice and the topics each CoS can offer advice on; ② On the matching between the Co's date-time preferences and the nominal availability of each CoS, the mediators of OHCS can mediate their mismatching. In order to facilitate the understanding of this scenario, in Fig. 2, we show a semantic discovery engine surrounded by a set of CoS and a client GUI. Each CoS exposes the functionality of arranging a meeting as a web service. The process that enables a Co to arrange a meeting with the most suitable CoS can be broken down in the following tasks:

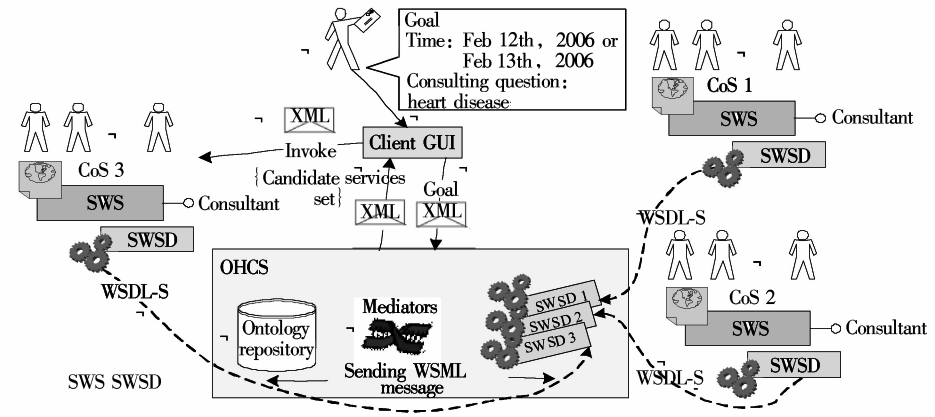


**Fig. 2** The working principle of OHCS

① The CoS provider and requester entities agree on the ontologies to use for modeling pathologies, drugs, advice services, date-time, etc. ;

② If they cannot agree on the use of a specific set of common ontologies, the definition of ooMediators isrequired. In this scenario, for instance, the CoS providers (Seoul time) and the requester (Beijing time) entities cannot agree on the use of a common date-time ontology;

③ Each CoS provider entity can register its web service descriptions into a semantic discovery engine. These descriptions include the medical issues the CoS deals with and the date—time intervals that the CoS is normally available;

④ A Co can discover the most suitable CoS by using a GUI. He/she may express his/her goal in terms of the available ontologies. For instance the Co asks for advice on a healthy consultation meeting on Feb 12th, 2006, or Feb 13th, 2006 (Beijing time);

⑤ The Co's goal is submitted to the semantic discovery engine;

⑥ With the aid of SII and SAI, the semantic discovery engine finds candidate services according to Co's goal among available semantic web services description (SWSD). During interaction processing, mediators are used to mediate possible mismatching (e. g., processing the difference between Seoul time and Beijing time);

⑦ The requester entity displays the results list to the Co;

⑧ The Co interactively selects an optimal CoS and uses a GUI provided by each CoS provider to invoke the service and book the meeting.

## 5 Conclusion

Really implementing semantic interoperability in SWS is convenient for connecting heterogeneous and distributed services providers and services requesters to fulfill complex business processes. Being compliant with the SSOA model, considering technological defects of previous models, we present a new set of interoperability mechanisms. By using ASM to model service interfaces and mediators to process four type of semantic heterogeneities, the mechanism not only efficiently resolves the problems of discovery and communication of services, but also is available for composition of services and semantic-based dynamic binding and execution.

## References

[1] Thatte S. Business process execution language for web services[EB/OL]. (2003-03) [2004-10-15]. ftp://www6. software. ibm. com/software/developer/library/ws-bpel. pdf.

[2] Kavantzas N, Burdett D, Ritzinger G. Services choreography description language version 1.0 [EB/OL]. (2004-05) [2005-12-20]. http://www. w3. org/TR/ws-cdl-10/.

[3] Grau B C, Parsia B, Sirin E. Working with multiple ontologies on the semantic web[A]. In: *Proceedings of the Third International Semantic Web Conference, LNCS*[C]. Hiroshima, Japan, 2004, **3298**: 205 - 219.

[4] Haas H, Brown A. W3C working group note 11[EB/OL]. (2004-09) [2005-12-10]. http://www. w3. org/TR/ws-gloss/.

[5] Krishna G. Abstract state machine and polynomial time logic[EB/OL]. (2005-05) [2005-12-30]. http://www. cse. iitk. ac. in/report-repository/2005/Y2158-CS397Report. pdf.

[6] Gurevich Y. *Evolving algebras*: *lipari guide*, *specification and validation methods*[M]. Oxford University Press, 1995. 9 - 36.

[7] Bruijn J, Lausen H, Krummenacher R. The web service modeling language WSML[EB/OL]. (2005-03) [2005-09-26]. http://www. wsmo. org/TR/d16/d16. 1/v0. 2.

# 语义 web 服务系统中互操作问题的研究

满君丰[1,2] 彭三城[1,2] 向剑伟[1] 胡永祥[1]

([1] 湖南工业大学计算机系,株洲 412008)

([2] 中南大学信息科学与工程学院,长沙 410083)

**摘要**:为了真正实现 web 上异构、分布的应用程序间的语义互操作,提出了一套有效的互操作机制. 该机制用抽象状态机模型化的服务交互接口和服务聚集接口来抽象化描述被调用的 web 服务实例的行为,使业务处理能够准确地进行任务指派,有效解决了服务双方的通讯和协作问题;给出适当的调解技术解决交互过程中的通讯和信息的不兼容问题,便于服务间的互操作、共享和集成. 通过一应用实例对该机制的工作机理和互操作实现进行了详尽的阐述.

**关键词**:面向语义服务的结构;抽象状态机;服务接口;互操作

**中图分类号**:TP312