WSDRI-based Semantic Web Service Discovery Framework

Xu Sun[1], YanLi Xu[2],MingRong Mao[3] and Ming Dong[4]
1 Renmin University of China, the School of Information, 100872 Peiking, xu2002261@163.com,
2 Renmin University of China, the School of Information, 100872 Peiking, xu_yanli@163.com.
3 Nanjing Normal University, Information department, 210042 Nanjing, mrmao@njnu.edu.cn
4 Renmin University of China, the School of Information, 100872 Peiking, dongming3699@126.com

Abstract. In the research of Web Service [1, 2], semantic information should be automatically discovered, selected and composed. These automations can make usage of Web Service easily. In this paper we propose a framework to facilitate the discovery of Web Service. In this framework, we use WSDRI (Web Service Discovery Information) to describe the semantic information. This framework which refers to client and Match Server is based on WSDRI. Then we evaluate the framework through the application on the internet to find that the framework is effective. Following this framework, it could be easy to discover the information of Web Service especially the semantic information.

1 Introduction

Now enterprise application begin to shift from single way to collaboration way. An application can be formed by web applications that base on the Browser/Server structure. And more and more applications come into distributed compute supported by many technologies such as CORBA, Java. Because of this environment and such technology, Web Service [1, 2] comes forth. Web service is a coupled component that can be programmed via any program language such as Java, C#. It exposes its interface on the Internet and some clients can request it. And Web Service is a standard including Hypertext Transfer Protocol, XML, SOAP, WSDL (Web Service Description Language) and UDDI[8]. If Web Service follows these technology standards, a client can call the Web Service via UDDI. There are many researches in this field. For example Wolfgang Hoschek proposed the web service discovery

Please use the following format when citing this chapter:

Sun, X., Xu, Y., Mao, M., Dong, M., 2007, in IFIP International Federation for Information Processing, Volume 252, Integration and Innovation Orient to E-Society Volume 2, eds. Wang, W., (Boston: Springer), pp. 271-281.

architecture [14]. It specified a small set of orthogonal multi-purpose communication primitives for discovery. And to integrate the information, he used XML data model. And then Ronan Barrett and Claus Pahl addressed service composing with a modeling approach, a non-intrusive decentralized interaction mechanism and a solution for dynamic deployment of the composition. Their novel approach combined a Model Driven Architecture using UML 2.0, for modeling and subsequently generating Web service compositions, with a method for achieving decentralized communication amongst services. They also provided a Web service based facility for enabling the dynamic deployment of compositions [15].

We think Web Service description should contain enough semantic information. This information is associated with some Web Service. When discovering a Web Service, this semantic information should be offered by a client. Via WSDA, at runtime, application can discover and adapt the suitable Web Service [14]. A simple description language SWSDL (Simple Web Service Description Language) is used to describe a Web Service [14] .But in this paper, we do not use another simple language to describe Web Service, and instead, we use a formalized method to figure out what information a client should offer, and how is Web Service discovered. We assume that we can use the present technology such XML to represent the related information about Web Service.

But the service providers should describe the Web Service using a certain language and register the services for some client can find it. At present the OWL [13,10] language should be considered. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. But we can use OWL-S [10]. OWL-S is ontology for services created by the Darpa Agent Markup Language (DAML) group (Martin et al., 2005). The ontology is broken into three parts: the Service Profile describes the capabilities of the service; Service Model describes how the service works internally; and Service Grounding describes how to access the service [6]. The OWL-S ontology is useful in that it provides a uniform mechanism for describing the semantics of a Web service. We use a tool to make us easy to construct the OWL files. We do not discuss the OWL language. We only offer a framework for discovering Web Service, and the framework is independent on some language, because we describe it using formal language.

In this paper we define some procedures and these procedures use the information of the semantic information to discover Web Service. These procedures run at client and Match Servers. The Semantic Information, which is named in paper as WSDRI (Web Service Discovery Routine Information), transfer between Match Servers. We apply the event-driven model to the framework; And the procedures we define in this paper, are all embedded in this model.

2 What is WSDRI

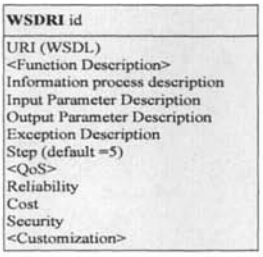
Semantic Web services are the emerging technology promising to become one of the future key enables of the semantic web. Now in this field, experts are researching many aspects of the semantic web, such as the standards of web service, the infrastructures, the match algorithms, the method to describe the semantic and so on. But Web Services are generally described using XML-based standards, namely

WSDL, UDDI, and SOAP [9,11,12]. In addition to these low-level standards, work is on-going to create standards that allow services to be combined into a workflow, e.g. WS-BPEL (Web Services — business process execution language), and also to define permissible message exchange patterns and contents, e.g. ebXML [14]. However, few of these standards provide a way to describe a Web Service in terms of explicit semantics. For a given service you maybe do:

(1) What kind of service it is (2) What inputs it requires (3) What outputs it provides, what needs to be true for the service to execute

The first of these requirements is partly addressed by UDDI, in that a category and human-readable description can be assigned to a Web Service in a registry to aid discovery [3, 5]. This provides only limited support for automated discovery since a computer will not understand the description or what the category means. The second and third of these requirements are partly addressed by WSDL in that XML tags can be attributed to inputs and outputs. A computer can easily match these, but again has no notion of their meaning or relationship to other pieces of data. Fundamentally, most of the hard work is left to the human user who must interpret the descriptions provided to the best of their abilities [4].

But the above discussion lacks the information to describe more features of Web Service. In this paper, we use Web Service Discovery Routine Information (WSDRI) (Fig.1) to conquer these disadvantages.



• **Fig.1.** The description of WSDRI

WSDRI id expresses the identification of the request for Web Service.

URL expresses Web Service's WSDL. If a client wants an explicit Web Service, this field should be assigned a value.

Function description expresses the function of Web Service. It includes: Information Process Description, Input Parameter Description, Output Parameter Description and Exception Description.

We think that there should be some criterions about Information Process Description (IPD), output description, input description, and exception description. And these criterions should also contain semantic information to help to match. In function description, we think IPD is very important and there should be an industry standard to specify IPD. It should be defined as Standard Information Procedure of Industry (SIPI) that is defined by industry or Standard Information Procedure of Business (SIPB) that is defined by some company. And a certain SIPI or SIPB can

describe an Information Process. Every service provider should had better follow these standards and provide more semantic information.

Step (default = 5) expresses that how many Match Servers the WSDRI should transfer in one direction.

QoS expresses the constraints of a Web Service's quality. It includes reliability cost, security and so on.

Customization expresses that the client can define some extra constraints about Web Service.

About the WSDRI, we can formalize the data structure, as follows:

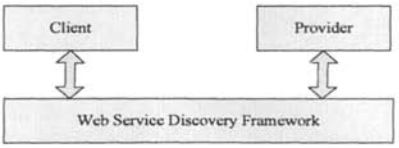
Need WebSevice:={ function description} ***with*** {non-function description}
{function description}:={
InformationProcessDescription:={semantic information};
OutputParam:={semantic information};
InputParam:={semantic information};
Exception:={semantic information};
}
Step:={5};
{non-function description}:={
Cost:={numeric};
Security:={encrypt standard};
Time:={***n*** seconds};
Customization:={non-function description}}

The service provider should offer enough semantic information to register its Web Service and this is out of the scope of this paper, so we assume there is lots of information in internet or managed by the Web Service Discovery Framework.

3 The Web Service Discovery Framework

3.1 General introduction of the framework

On the Internet, there should be several hosts which run match software. These hosts can use WSDRI to match the semantic information. We name these hosts as Match Server. Some client sends WSDRI and the framework can return a set of Web Service to this client (Fig.2). The Match Server is linked as a chain (We will talk the match server chain at **3.2**):



• **Fig.2.** usage of Web Service Discovery Framework

For Match Server to use WSDRI to discover the Web Service, we define such processes as follows:

SendWSSemantic:

(1) InParam: WSDRI (is identified by requestor's URL and the WSDRI id)(2) OutParam: None(3) Called by client

CombineResults:

(1) InParam: ArrayList (two results)(2) OutParam: an array list of Web Service in which there is no the same element.(3) Called by client

MatchSemantic:(1) InParam:WSDRI, WSDLs (a set of Web Service)(2) OutParam: None(3) Call by Match Server

SendNextChainServer:(1)InParam: inWSDRI (a copy of WSDRI), outNextSetofWS (a set of Web Service)(2)OutParam:None(3)Call by Match Server

SendSetofWS:(1) InParam: a set of Web Service; WSDRI (2) OutParam: None(3) Call by Match Server

We use Event System to invoke this function. The event is a Message Object. We use the kind of object to decide which handler is called. So we define such types:

WSSMsg (Web Service Semantic Message) expresses that in client the queue of Web Service Semantic Description is not empty, and then the handler SendWSSemantic is called.

WSCMsg (Web Service Combine Result Message) expresses that in client the two results received and the handler CombineResults is called.

WSMMsg (Web Service Match Semantic Message) expresses that in Match Server the WSDRI is received and the match software will call the handler MatchSemantic .

WSSSMsg (Web Service Send Set Message) expresses that if the WSDRI get the end of Web Service Chain (WSC) or the ***Step*** field of the WSDRI is reduced to zero, the handler SendSetofWS will be called.

We assume that in client and Match Server there are some queues.

In client there are:

1. Semantic Request Queue stores the WSDRIs. In client every request for a Web Service is associated with a WSDRI.

2. Result Queue expresses that one WSDRI will generate two results and when the two results return, they will be saved in this queue. Every element is divided into two units to save the two results.

In Match Server there are:

1. Match Semantic Queue expresses that WSDRI which was received from a client or another Match Server will be saved in this queue.

2. Final Result Queue expresses that when the ***Step*** field of the WSDRI is reduced to zero, the set of Web Service in current Match Server will be saved here and waits to be sent to the client.

3.2 Procedure definition

The framework works based on the mentioned procedure at the above sections. So we now define the procedures in this section. First we define the SendWSSemantic () procedure and this procedure is used in client. The client uses this procedure to send the WSDRI to a default Match Server.

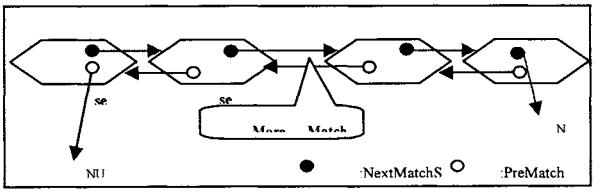
SendWSSemantic (inWSDRI):={
IF validate (inWSDRI) =TRUE
THEN SYSTEM . send (inWSDRI)
Else
{require More Details}
}

WSDRI should have some essential elements. So we call function validate () to confirm if inWSDRI (inWSDRI is a copy of WSDRI) is integrity. We think such essential elements are: function description, output description, input description. Finally we use a system call send () function via a default Match Server URL to send WSDRI to Match Server.

Every Match Server receives WSDRI, the function as follows we define should be invoked.

MatchSemantic (inWSDRI, inPreSetofWS){
IF inWSDRI. ***Step***<MaxStep
THEN set inWSDRI.***Step*** inWSDRI.***Step***-1
Set tmpSetofWS:={
Select WSDL from ontology Server linking to the Match Server where Semantic Server SUPER inWSDRI
}
Set outNextSetofWS :={ UNION (tmpSetofWS, inPreSetofWS)}
IF inWSDRI. ***Step***<0 OR (NextMatchServer is NULL OR PreMatchServer is NULL)
THEN SendResultModule (inWSDRI.URL , outNextSetofWS)
ELSE
SendNextChainServer (inWSDRI, outNextSetofWS)
}

Before discussing the above function, we first introduce the Match Server Chain (Fig.3).



• **Fig.3.** The interior structure of the match server

But in this framework we assume that all Match Server are inter-linking like as a bidirectional link-table(Fig.3) and we use PreMatchServer and NextMatchServer to identify the preceding and next server.

In function MatchSemantic (), the Match Server first check the Step field and determine if the WSDRI should be completely processed; and if the result can be send client by calling the SendResultModule. About modules we will discuss at the next chapter.

In MatchSemantic () procedure, procedure SendNextChainServer () is called. Because the Match Server is a chain, and the ***step*** is not zero, the procedure will be called. The definition is following as:

```
SendNextChainServer (inWSDRI, outNextSetofWS) {
IF FROM(inWSDRI)=PreMatchServer
THEN SYSTEM . SendNext(NextMatchServer , inWSDRI , outNextSetofWS)
   ELSE
       IF FROM(inWSDRI)=NextMatchServer
THEN SYSTEM . SendNext(PreMatchServer , inWSDRI , outNextSetofWS)
ELSE {
SYSTEM . SendNext(NextMatchServer , inWSDRI , outNextSetofWS)
         SYSTEM . SendNext(PreMatchServer , inWSDRI , outNextSetofWS)
}
}
```

SendNextChainServer () should decide which URL the WSDRI should be send to. In Match Server chain , if a WSDRI is received, and its source URL is PreMatchServer, the WSDRI should be send via the NextMatchServer, and otherwise via the PreMatchServer. And if the WSDRI is from a client, the WSDRI will be send via PreMatchServer and NextMatchServer. FROM () is to fetch the source URL.

When in a Match Server Chain some a Match Server receives a WSDRI from a client, the WSDRI will be send to two servers, one is PreMatchServer and the other NextMatchServer. And so the two directions will be generated two result sets and the client will receive them. And the following procedure is necessary. It use UNION operation to merge two returned results. The procedure CombineResult () is called when the WSCMsg message happens.

```
CombineResult (firstResult, secondResult, outOptionalResult) {
Set outOptionalResult: =UNION (firstResult,secondResult)
}
```

3.3 Module definition

As we mentioned in the previous sections, we think that the framework is based on the message-driven model. In this section we will discuss the relative aspects about the model.

First, in this section we only use one procedure as follows we define:

```
SendSetofWS (URL, inSetofWS){
SYSTEM. Send (URL, inSetofWS)
}
```

The procedure is called when the final result set is formed, and the message WSSSMsg generates in Match Server. But before this the final result should be save in the Final Result Queue. This procedure is simply calling the System function Send ().

We think that there should be some modules that generate event and set some environment variables to offer the executing environment for the event handler. So we define some Modules run at client and Match Server:

1. ClientModule:This module run in client and generate event: WSSMsg (Web Service Semantic Message), MSCMsg (Web Service Combine Message). And it can operate Semantic Request Queue and Result Queue.

2. MSModule:This module runs in Match Server and generate event: WSMMsg (Web Service Match semantic Message). And it can operate Match Semantic Queue. This module calls Send Result Module.

3. SendResultModule:This module runs in Match Server and generates WSSSMsg. And it can operate Final Result Queue.

Then we formalized the three modules as follows:

ClientModule
Begin
Suspend on PORTxxx until WSDRI ***is listened*** {
Add WSDRI ***to*** SemanticRequestQueue
Generate WSSMsg
}
Suspend on PORTyyy until Set of Web Service ***is listened*** {
Add Set of Web Service ***to*** Result Queue
IF two result sets that refer to the same WSDRI return ***Generate*** MSCMsg
}
End
MSModule
Begin
Suspend on PORTzzz until WSDRI is ***listened*** {
Add WSDRI ***to*** Match Semantic Queue
Generate WSMMsg
}
End
SendResultModule
Begin
Add Set of Web Service ***to*** Final Result Queue
Generate WSSSMsg
End

And if some event is generated, we use the macro ***BEGIN_ EVENT*** and ***END_ EVENT*** to determine which procedure should be invoked.

In client:

BEGIN_ EVENT
WSSMsg ***Associated with*** SendWSSenmatic
WSCMsg ***Associated with*** CombineResult
END_ EVENT

In Match Server:

BEGIN_EVENT
WSMMsg ***Associated with*** MatchSemantic
WSSSMsg ***Associated with*** SendSetofWS
END_EVENT

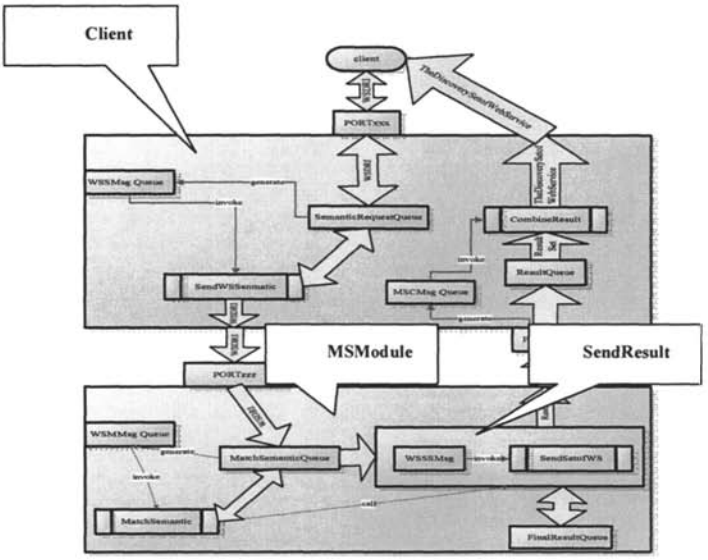
4 Conclusion and Future work

4.1 Conclusion

This paper has presented the Web Service discovery framework. We use WSDRI to describe the semantic information about a Web Service including the input parameter, output parameter and some of the function and non-function information, conquering the deficiency of the WSDL and the UDDI.

This framework integrates the main features of Web Service. The semantic information of Web Service is published in the framework by the service providers. So the client only provides the semantic information like a natural language but not like the program interface, and the framework can complete the discovery of Web Service. In client eyes, discovery of Web Service is easier via WSDRI.

About our work at the previous mention, we conclude our research and the Web Service discovery framework can describe as following chart (**Fig.4.**):



• **Fig.4**. The detail structure of the Web Service Discovery Framework

If a Client has a Web Service requirement and the WSDRI will be generated. At this time event WSSMsg will be generated and WSDRI will be saved in a queue, Semantic Request Queue, and the event handler SendWSSemantic will be called.

When a Match Server receives a WSDRI, the event WSMMsg will be generated and the received WSDRI will be saved in Match Semantic Queue. In the Match Server, because WSMMsg occurred, the corresponding handler MatchSemantic will be called. But if the value of ***Step*** field of WSDRI is reduced to zero, the current set of Web Service will be the final result set and this set will be sent to the responding

client. When the client receives two sets of Web Service that is identified by the same WSDRI id, the event WSCMsg will be generated and the two result sets will be saved in a queue Result Queue. Because the event WSCMsg occurred, the handler CombineResults will be called. Then the client can get a set of Web Service from the framework works in the Internet.

4.2 Future work

Though the framework is effective, there are some shortcomings. In order to complete the framework fully we should do the next in the future.

1. We think that in Web Service how to process the data is very important and that is the most part of the function. We will embark on research the SIPI and SIPB. This can describe the meaning of semantic information fully, as a part of the WSDRI. Now the description of that is insufficient.

2. We will develop a system or prototype to implement the framework and try to find more dimensions about WSDRI. We think the technology of developing this framework is not difficult. This prototype will follow our framework. Many program languages all support the development of module of the framework, such as Java. But considered the efficiency of the framework we will use the C++ language.

3. We will develop some algorithm to evaluate the Web Service that be returned from the Internet and then we will make the auto-selection and the auto-composition coming true. An auto-selection and auto-composition is considered the next most important work in the research of web service. If these come true, the web application will be produced automatically.

References:

1. Max Haustein, Klaus - Peter L#hr. JAC- Declarative Java Concurrency, Concurrency and Computation: Practice and Experience, 2006, pp.519~546.
2. Gerald C. Gannod, John T. E. Timm, Raynette J. Brodie. Facilitating the Specification of Semantic Web Services Using Model-Driven Development. IDEA Group Publishing, Volume 3(3), 2006, pp.61-81.
3. Jyotishman Pathak, Neeraj Koul. A Framework for semantic Web services discovery. ACMWIDM, 2005.
4. M. Altenhofen, E. B¨orger, J. Lemcke. An execution semantics for mediation patterns. In C. Bussler, D. Fensel, U. Keller, and B. Sapkota, editors, Proc. of 2nd WSMO Implementation Workshop WIW'2005.
5. A. Barros, E. B¨orger. A compositional framework for service interaction patterns and communication flows. International Conference on Formal Engineering Methods (ICFEM 2005), volume 3785, 2005, pp.5–35.
6. A. Barros, M. Dumas, P. Oaks. A critical overview of the web series choreography description language (WS-CDL). White paper, 24th of January 2005.
7. D. Skogan, R. Gronmo, I. Solheim, "Web service composition in UML", In: Proceeding of the 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC'04), pp. 47–57, Sep, 2004

8. L. Li, I. Horrocks. A software framework for matchmaking based on semantic web technology. In Proc. of the Twelfth World Wide Web Conference, 2003.
9. Bijan Parsia. Semantic Web Services. Bulletin of the American Society for Information Science and Technology, May, 2003, pp.12-15.
10. Methmet Sayal, Akhil Sahai, Vijay Machiraju, Fabio Casati. Semantic Analysis of E-Business Operations. Journal of Network and Systems Management, May 2003, pp .13-37.
11. Maedche A, Staab S. Services on the Move – Towards P2P-enabled Semantic Web Services. In: Proceedings of 10th International Conference on Informa- tion Technology and Travel & Tourism, ENTER 2003, Helsinki, 2003.
12. C. Goble, D. D. Roure. The Grid: An Application of the Semantic Web. ACM SIGMOD Record Volume 31(4), December ,2002, pp.65-70.
13. P.H. Alesso, C..F. Smith, Developing Semantic Web Services, A K Peters ltd, Wellesey MA, Canada, Date, 2004, pp.165-272.
14. Wolfgang Hoschek . The Web Service Discovery Architecture . CERN IT Division . European Organization for Nuclear Research .
15. Ronan Barrett, Claus Pahl, Lucian M. Patcas, John Murphy. Model Driven Distribution Pattern Design for Dynamic Web Service Compositions ICWE'06, July 2006, pp.11-14.