一种统一的服务描述语言分析与设计①

司华友 1,2,3　张　力 1,2,3　陈　钟 1,2,3,4　胡建斌 1,2　倪宇林 3,4

(1.北京大学 信息科学技术学院 北京 100871;2.高可信软件技术教育部重点实验室 北京 100871;

3.北京大学 ACOM 金融信息化研究中心 北京 100080;4.北京大学 软件与微电子学院 北京 102600)

Analysis and Design of a Unified Service Description Language

Huayou Si [1,2,3], Li Zhang [1,2,3], Zhong Chen [1,2,3,4], Jianbin Hu [1,2], Yulin Ni [3,4]

(1. School of Electronics Engineering and Computer Science, Peking University, Beijing 100871;

2. Key Laboratory of High Confidence Software Technologies Ministry of Education, Beijing 100871;

3. PKU-ACOM Financial Information Research Center, Peking University, Beijing 100080;

4. School of Software and Microelectronics, Peking University, Beijing 102600)

Abstract: During the recent years, Service-Oriented Computing (SOC), as a new computing paradigm, has been widely accepted in academies and industry. More and more service-oriented computer system analysis and design methodologies have been proposed. It is believed that if a service-oriented methodology is based on compatible concepts and description methods between the application domain and the system responsibility, it would take more advantages. So, in this paper, a Unified Service Description Language (USDL) is suggested based on our understanding of service and its attributes to unify description of web service and general service in application domain. With a demo in DELCCA project, USDL is asserted to efficiently support service oriented analysis and design.

Key words: service; web service; unified service description language (USDL); SOAD

1 Introduction

In the past few years, in the computer domain, software engineers began to adopt “service” concept which came from Economics to develop and deploy applications. Now, Service-Oriented Computing (SOC) is widely accepted, which represents a new computing paradigm. SOC will change the way we develop and use software [1].

SOC is emerging for building and maintaining applications in a cost effectiveness way. However, there is a considerable amount of gaps between the promises of SOC and the maturity of service engineering methodology. The main cause of the gaps is the lack of effective analysis and design methods[2]. That is to say, there is a great demand on effective service-oriented analysis and design (SOAD) methodology[3,4].

① Supported by the National Natural Science Foundation of China under Grant No.60773163

Analysis and design inevitably link their problem domain and system responsibility, and describe them, which must apply the domain's concepts. In the process, "service" is a central concept that is used in both problem domain and system responsibility. So it should import problem domain into system straightly. Though there have been several methods to describe service in domain or in computer, yet analyzer and designer have few practicable methods to describe service in both domains effectively[5]. In fact, a good SOAD methodology must be based on a good and accordant description method to application domain and computer concept, such as web service [6].

How to describe a service and how to explain the different service deliveries are questions which bother the analysis for a long time[1,7]. Based on survey of some service definitions and ideas about service essence, this paper proposes an accordant service description method, which will provide strong points to SOAD. In Section 2 of this paper, the ideas the method bases on are discussed. In Section 3, the USDL method is defined. In Section 4, the paper analyzes traditional description of web service, i.e. wsdl, how to be translated into the method. In Section 5,authors use the method to describe the DELCCA project roughly to assert that it efficiently supports SOAD. And in the last section, the paper discusses the characters of the method and intending research.

2 Ideas USDL Bases on

In this section, some ideas are listed as follows, which will be used as bases to design USDL to describe service which includes web service in computer domain.

2.1 Everything is service

Any interaction between two entities can be regarded as a service process. One entity can be regarded as provider of the service process, and the other is user of the service. The interaction can be a collaboration process of two pieces of program, or the course of conversation between two people. They can be abstracted as a service process. So, every collaboration or even average interaction is regarded as a service process. Service can be modeled any process between two entities.

2.2 Service is always provided by service network

When a service entity provides service for a user, it maybe needs several other services. In fact, in our society, there are few services which can serve its customer independently[8]. In contrary, when serving its customer, almost all services must consume some other services as a customer. For example, we all know, when going to shopping, the shop provides services for us. At the same time, the shop must need some other services, such as manufacturer, employee, electric power plants, and so on. The relationship of dependence can be considered as service network[9,10]. Especially, in order to provider a given service, several service entities must collaborate based on the relationship of dependence. The relationship among the several service entities can be described by diagram, such as Fig.1. The diagram here can be named service network. In the diagram, the arrowheads point to service receiver.

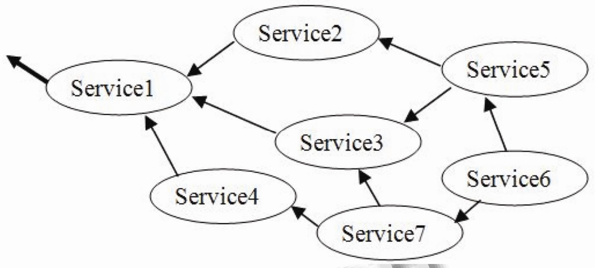

Fig.1 Service network

2.3 Service is hierarchical

In order to provider a given service, as a user, a service entity always uses some other services. The services used always are hidden by the user services, which can not be found or seen by the end user. For instance, when we receive the service of a mobile telephone, we do not care about the big system behind the set, such as signal tower which send signal for mobile telephone. Their functions are encapsulated by the interface services entity, which interacts directly with end user. About the given service, which is the function-imited service entity, such as service1 in Fig.2, can represent the total service network function. So, we denote the representation as an abstract new service, such as service0 in Fig.2. And we adopt hierarchical structure to express this kind of relationship. In itself, service0 is composite service.

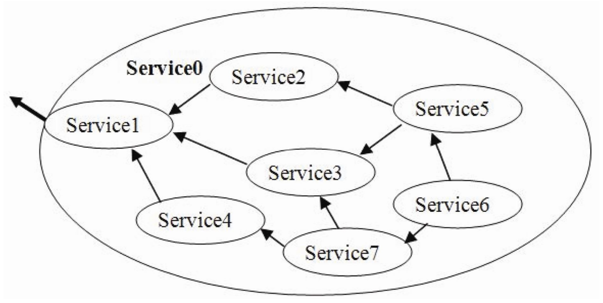

Fig.2　Service hierarchical structure diagram

2.4 Service should be preexistent

Service is designed for a special application, even for no purpose. But, no one knows how it will be used in other way in the future. No one knows that its user how to be used too. Maybe, in the future, it or its user is found dynamically, and used in a given collaboration which is not foreseen.

So, in substance, user, which must be a person or another service entity, is clever than the service. It must know how to use the preexistent service. Service is preexistent, this is to say, service entity is designed not only for a given or intending application, but also for potential, unborn and unknown application. Unborn application usually is established based on those preexistent services.

So, a service does not and can not care about how to be used in the future. It only cares about "what I can do, and how I will do", which are just that we describe about a service.

A user will find dynamically a service based on what it can do, and begin to design a collaboration process to interact with the service based on how it will do it. "How to do it" is like a mirror. When the user set eyes on the mirror, the clever user will reflect its action in the future interaction with the service. So the collaboration rule will be worked out. The method which describes service bases on the principles.

2.5 "Serve" is different from service

"Serve" is an interactional process between provider and user, which can not depart from a given provider and its user. In the process, "serve" realizes value[11]. Value lies in the process. If we have only the service ability that can be called service entity or service system, we can not achieve values. Each process in which "serve" is realized is particular. The process is provided by service, i.e. service entity or service system. So, when describing a service, the interaction process needn't be described, since it is not existent. It is only existent when the "serve" occur. The thing which needs describing is service logic or service rule, which will generate a service process according to particular requester or surroundings.

3　Service Description Method Definition

Based on the concepts that are discussed as above, a service can be described from two aspects. The first aspect is service interactional logic, which can be called behavior logic of a service. From service own point of view, behavior logic describes service interaction rules which is regarding to service interactive interface. Based on the interaction rules, service expresses what it can do and which values it can provide. Basing on the interaction rules too, the potential user judges whether to adopt the service and how to interact with the service when their collaboration occurs. Of course, the designer can use to design a perfect service system.

The other aspect is service organization logic, which describes the service how to work. Here, we adopt the two concepts of service network and hierarchical structure, which are discussed just as above.

So now, we design a service description or definition as follows:

Service =< behavior, organization>

3.1 Organization logic definition

In the two-tuple, the first element behavior repress-ents the service interaction logic, and the second element organization for organization logic.

organization =<Services, PR, SuperS>.

The organization is defined as a tuple too. It is a three-tuple. The first element shows which services are involved in the big service objective. It is set, which include the services involved, as follows:

Service={service| services is involved}

PR=<service1, service2>

PR is a binary relationship, which represents that service2 provides service for service1. And:

service1, service2∈Services

If organization =NULL, it is equivalent to Services =Φ.

When service is an atomic, or service is provided by an entity or a system which do not need analyzing, the service organization should be defined NULL.

Obviously, the tuple<Services, PR>is an analogous Partial Ordered Set. It is not a real Partial Ordered Set, because it can meet three principles of Partial Ordered Set:

<service1,service1> is not an element of PR, it must be an element in Partial Ordered Set.

Obviously, if <service1,service2> ∈ PR, then <service2,service1> is not an element of PR. It is common both in the analogous Partial Ordered Set and in a Partial Ordered Set.

if <service1,service2>,< service2,service3> ∈ PR, then <service1,service3> is not always an element of PR.

So, relationship set < Services, PR > is only an analogous Partial Ordered Set. In fact, < Services, PR > is unnecessary defined as a perfect Partial Ordered Set. But, now the organization set can be described by Hasse Diagram too.

SuperS is a set which has the attribute that SuperS⊆ Service. The elements of the set are maximum elements in the analogous Partial Ordered Set. Maximum elements in the Set refer to the services that the upper service's user sometimes touches with them directly. This is to say, the lower services in SuperS which encapsulate by the upper service have the opportunity to represent the upper to interact with the user.

Naturally, organization logic can be described with a kind of ontology language, such as OWL-S.

3.2 Interaction logic definition

In the definition of service, the first element describes the service behavior logic, which is tagged as "behavior". "behavior" is defined as a triple-tuple, as follows:

behavior=<InteractionsWithUser, firstAction, lastActions>

In the behavior set, InteractionsWithUser is a set. The set contains lots of action sections which will be adopted when the service serves for a user. firstAction is a element in InteractionsWithUser. And it is a special action, which is an initial action that must be adopted when the service begins to serve. lastActions is a set too. It includes always more than one actions. When the service adopts one action of lastActions, it is means that the serving process is over. For the sake of convenience, the concepts, such as action and nextActions, are defined as follows:

action∈InteractionsWithUser

firstAction∈InteractionsWithUser

lastAction∈lastlActions? InteractionsWithUser

nextActions⊆InteractionsWithUser

Action is element in the InteractionsWithUser. It is a basal element of behavior logic. Based on special environment, the service chooses an action to deal with the user until the action chosen is an element in lastAction. The chosen actions and the action the user reflects make up of a process, which is specific "serve" provided for its user.

Here, the action is defined as quarter-tuple. The first element describes the action type. The second element represents things or messages which are exchanged between user and the service when the action occurs. "results" represents a result set which contains all results and effects of the action. "rules" shows the logical relationship between the action and other actions under the given result.

action=<actionType, paraThings, results, rules>

actionType ∈ {acquire, send, do, wait, close, initiate}

"acquire" and "send" type action can be used to reflect the receiver's behavior, since if the service begins an acquiring action or accomplishes a sending action, the receiver must take an corresponding action. And "wait" action can be regarded as a kind of acquire action. So, it can reflect the receiver's behavior too. The "do" action represent a service own action.

lastAction and firstAction can be detailed generally as follows:

lastAction=< close, Null, Null, Null>

firstAction=<initiate, Null, Null, {nullRule}>

"rules" describes the logical relationship between the action and other actions. A rule is binary-tuple too. If a rule belongs to an action, the rule first element is a result belongs to the action. And the next element represents which is the next action the service will adopt under the current result. The formulae and concepts are defined as follows:

rules⊆Rules

rule ∈ rules

rule|thisaction =< result|thisaction, nextActions>

nullRule =< null, null, {action}>

nextAction=action ∈ nextActions

paraThing⊆paraThings

paraThing∈serviceDomainOnto

"behavior" element in a service can be described with OWL-S and SWRL(A Semantic Web Rule Language).

So, now a theorem can be conclude obviously:

behavior|service=ΣSuperS behavior

It means that a service behavior logic is equal to summary behavior logic of services in the SuperS in the service organization logic.

4 An Example of Translation from Traditional Methods into the Method

Similarly, the languages, such as WSDL、OWL-S, are used to describe web service. However, they only describe the web service interface, which includes output and input, sometimes, includes effects and results. Even though web service is a composite service how is process in essential made up with other web services described by OWL-S. So, these languages can not describe the complex interaction between the service and its user.

But web service is one kind of service which unifies the service concept in computer memory and in real world. For observing the world from the interactional and dynamic aspect, service can be used to model and unify any interactional behavior. So, we can use the description method to describe web service. Therefore, it can be translated into the description that is given above. This is a segment adopting from wsdl document as follows:

```
  <interface    name = "reservationInterface" >
     <fault name = "invalidDataFault"
            element = "ghns:invalidDataError"/>
        <operation name="opCheckAvailability"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
          <input messageLabel="In"
            element= "ghns:checkAvailability" />
          <output messageLabel="Out"
            element="ghns:checkAvailabilityResponse"
/>
          <outfault ref="tns:invalidDataFault"
            messageLabel="Out"/>
    </operation>
</interface>
```

This wsdl segment can be translated into our description as follows:

theService=< ourBehav, ourOrgani>

ourOrgani =Null

ourBehav =<theIWithUser, thefirstAction, thelastActions>

theIWithUser={acquireAction,sendAction,thelastAction, thefirstAction }

thefirstAction=<initiate, Null, Null, {theNullRule}>

theNullRule=< null, {acquireAction }>

thelastAction=< close, Null, Null, Null>

acquireAction =< acquire, paraIn, results, rules>

paraIn="ghns:checkAvailability"

results={getParaIn}

rule=<getParaIn, sendAction >

sendAction =< send, paraOut, results, rules>

paraOut∈{"ghns:checkAvailabilityResponse", "tns:invalidDataFault"}

results={ paraOut}

rule=< this, paraOut, thelastAction >

5 The Method Used in DELCCA as a Demo

DELCCA is an ongoing project deploying an integrated location-aware service system for passengers, airlines and airport management. DELCCA Research Workshop is made up of Peking University and IT University of Copenhagen. The intending system will adopt web service technology to analyze, design and implement. When deployed, the system will detect passenger location in airport at any moment, and then inform passengers what to do just now to avoid delaying his or her airline. It is expected to be operational in 2009 in Copenhagen Airport, using RFID and Bluetooth detection that tracks mobile phones, passenger badges and trolleys.

Now, in the project analysis phrase, we try to adopting service viewpoint to analyze the project requirement and describe the service found in the project in USDL.

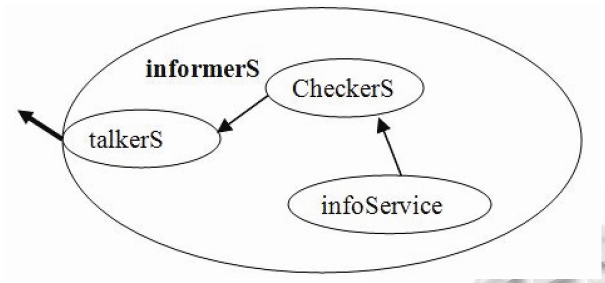

Fig.3 Service identification in whole project scope

A very simple and rough service analysis in DELCCA project based on the description is listed as fellows:

First, the project is regarded as the most upper service, which is called "informerS". It directly serves passengers. It can detect passenger location real time. If needing, it informs passengers something. The upper service is realized by three other services, according to intending technology adopted, which are talkerS, CheckerS and infoService.

"informerS" represents the intending whole project interaction with passengers in airport. It is described with USDL as follows:

informerS=<informerSBehav, informerSOrgniz>
informerSOrgniz=< informerServices, informerPR,
informerSuperS >
informerServices ={talks, checkers, infoService}
informerPR={<talkerS,checkS>,<checkS,infoServic
e>}
informerSuperS={ talkerS}
informerSBehav=<informerInteractions,
informerfirstAction, informerlastActions>
informerfirstAction=<initiate,Null,Null,{nullRule}>
theNullRule=<null, null, { callAction }>
callAction=< acquire, callMessages, callResults,
callRules >
callResults={getMessage,!getMessage}
callRules={<getMessage,showAction>,<!getMessage,
callAction >}
showAction=< send, showMessages, showResults,
showRules >
showResults={sendshowMessage}
callRules={<sendshowMessage,informerlastAction>}
informerlastAction=<close, null, null>

"talkerS" represents a service function of electronic device in passenger hand to provide information service directly. And it is an only element in informerSuperS in informers. So:

talkerS=< talkerSBehav, talkerSOrgniz>
talkerSBehav= informerSBehav

Because informerSuperS={talkerS}, and talkerS is the only element in set of informerSuperS, we can believe "talkerSBehav= informerSBehav" is right.

talkerSOrgniz=Null.

Because talkerS is a automible mobile telephone or some electronic set, it can provide the service all by itself. We do not need analyze it.

Now, as for CheckerS, if we believe that it which provides its service is depended by a service network, we should analyze the service network deeply. But, now we can regard it as a self-governed electronic product. We will not analyze its service network deeply. We only describe it as follows:

CheckerS=< CheckerSBehav, CheckerSOrgniz>
CheckerSOrgniz=Null.
CheckerSBehav=<CheckerInteractions,heckerfirstA
ction,
CheckerlastActions>
CheckerfirstAction=<initiate,Null,Null, {nullRule}>
nullRule=<null, null, { checkAction }>
checkAction=<acquire,checkMessages,checkResults,
checkRules >
checkResults={getManyTalker,getNoneTalker}
checkRules ={< getManyTalker, judgeAction>,
<getNoneTalke, CheckerlastActions >}
CheckerlastAction=< close, Null, Null, Null>
judgeAction=< do, showMessages, judgeResults,
judgeRules >
judgeResults={ManyTalkerNeed, NoneTakerNedd}
judgeRules={<ManyTalkerNeed, sendAction>,
< ManyTalkerNeed, CheckerlastActions >}

sendAction< send, sendMessages, sendResults,
sendRules >
sendResults={sendSucceed, sendFailure}
judgeRules={<sendSucceed,CheckerlastActions>,< sendSucceed,
sendAction >}

Then, if needing, we can describe infoService’s service behavior and its oganization or by the way indepth.

6 Conclusion and Future Works

In the paper, a kind of service description language is put forward based on listed ideas of service and its attribute, which is called USDL, i.e. unified service description language.

First, the method USDL unifies description of web service and general service concept. So, general service concept and web service now are equally treated. Because general service concept always is adopted in analysis phase and web service is in design phase, USDL provides a consistent concept and method to describe service, i.e. a thing important in service oriented system. Based on the idea that every interaction can be modeled as service, SOAD will be a new angle and method to understand and implement intending system. In the method, USDL will play an important role.

USDL takes the standpoint of service self. It does not describe the interactive processing between service and its customer, since we believe interactive processing exists in the moment when interactive processing occur, it is specific and instantaneous. So we can not describe it. It does not describe the processing procedure of the service, since we do not need. Based on the description, designer knows the processing procedure of the service how to do. It does not describe the processing procedure of the user too, since it is other’s matter, we do not care about. What it describes is service behavior logic. It firstly describes every possible action of the service and results that will produce once the action occurs.

USDL is also powerful. The concepts in interactive processing, such as synchronization and asyn, can be realized in its description logic. It can be translates into owl description, which will provide potential ability to service dynamic collaboration based on reasoner.

Naturally, it is not perfect. It should be put in practice in more real project. Good SOAD accordance with it should be designed too.

References

1 Tsai WT, Wei X, Paul R, Chung JY, Huang Q, Chen Y. Service-oriented system engineering (SOSE) and its applications to embedded system development. SOCA, 2007,1:3－17.

2 Chang SH, Kim SD. A Systematic Approach to Service-Oriented Analysis and Design. Product-Focused Software Process Improvement-8th International Conference, PROFES. Proceedings.v 4589 LNCS, 2007:374－388.

3 Sigh M, Huhns M. Service-Oriented Computing:Semantics, Processes, Agents. Wiley, Chichester, 2005.

4 Kambhampaty S. Service Oriented Analysis and Design Process for the Enterprise. 7th WSEAS International Conference on Applied Computed Science, Venice, Italy, 2007.

5 Jamshidi P, Sharifi M, Mansour S. To Establish Enterprise Service Model from Enterprise Business Model. 2008 IEEE International Conference on Services Computing. Hawaii, USA, 2008.

6 蔡维德,白晓颖,陈以农.浅谈深析面向服务的软件工程.北京:清华大学出版社, 2008:84.

7 Zhang L, Zhang NY, Chen Z. A Reference Service Description Framework. Journal of Harbin Institute of Technology, 2008,09(15):88－93.

8 Tien JM, Berg D. A Case for Service Systems, Engineering. Journal of Systems Science and Systems, Engineering, March 2003,12 (1):13－38.

9 Wang ZJ, Xu XF, Mo T. Service Architecture: High Level Descriptions of Service System. Journal of Harbin Institute of Technology, 2008,09(15):7－12.

10 Kwan SK, Min JH. An Evolutionary Framework of Service Systems. Journal of Harbin Institute of Technology, 2008,9(15):1－6.

11 Van Nuffel D. Towards a Service-Oriented Methodology: Business-Driven Guidelines for Service Identification. On the Move to Meaningful Internet Systems2007:OTM2007Workshops,2007,11:29－303.