WEB SERVICE RETRIEVAL BASED ON ENVIRONMENT ONTOLOGY

Budan Wu and Zhi Jin
Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing, China

Keywords: Web Service Description, Web Service Retrieval, Environment Ontology, Semantics.

Abstract: Existed descriptions of Web Service are in syntax level, and still relies on people to understand and decide what a Web Service can do, which encumbers automatic and dynamic Web Service retrieval. In this paper, we propose environment ontology as the base of describing semantic of Web Services, whose resource and state transitions reflect Web Service's behavior. We construct the description format based on environment ontology and put forward a road map approach towards automatic and dynamic Web Service retrieval.

1 INTRODUCTION

The booming of Web Services requires Web Services encoded in a description form understandable to computer. With this goal, the industry put forward WSDL(E.Christensen et al., 2001) and BPEL(Fu et al., 2004) which are standards for Web Services definition and orchestration. However, these standards are on syntax level using ports and interface to invoke Web Service functions. A first step toward this problem is to develop formal languages and inference mechanisms for representing and reasoning with core Web Service concepts.(Paolucci and Sycara, 2003) Thus, in parallel with those efforts introduced above, the Semantic Web(Berners-Lee et al., 2001) community has been developing languages and computing machinery. Efforts include RDF(Gutierrez et al., 2005), OWL-S(de Vergara et al., 2005) and IRS-II(Motta et al., 2003). Unfortunately, these efforts show limitations in automatic Web Service retrieval in practice. Motivated by accurately depicting Web Services in computer-understandable form, and specifying constraint on action sequence, state transitions, we use environment ontology with richer semantics to depict the specific capability of Web Services and put constraints on interaction sequences, states transitions and other related. The classic formula(M.Jackson, 1997) in requirement engineering expresses the importance of environment(M.Jackson, 2001). Deriving from it, environment changes reflect Web Service behavior. When both Web Service capability and requirement are mapped to the same environment ontology, a matchmaking can be developed between them automatically. The originality of this approach compared to other methods is that we use environment ontology to describe Web Services in formal semantic level and put forward a procedure of automatic Web Service retrieval based on environment ontology.

The paper divides into 5 sections. After the introduction, environment ontology and an online selling domain ontology are explained in section 2. Based on environment ontology, section 3 shows Web Service description method. In succession, Web Service retrieval is expounded in section 4. Finally, a conclusion is made.

2 ENVIRONMENT ONTOLOGY

2.1 Top-Level Environment Ontology

Top-level ontologies provide a set of basic relations and the classes to express the constraints on their use. It is represented by: $TO := < TC, TR, H^{TC}, Rel, A^{TO} >$

- TC and H^{TC} are concepts explained in Table 1.

Wu B. and Jin Z. (2007).
WEB SERVICE RETRIEVAL BASED ON ENVIRONMENT ONTOLOGY.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 382-385
DOI: 10.5220/0001275403820385

The first section of the table represents the resource ontology of the Web Service, the second represents the interaction ontology between Web Services and environment entities.

Table 1: Some Meanings of the Representation Concepts.

Environment Entity	*entities which a system will*
Atomic Entity	*interact with an individual entity*
State Machine	*a direct graph*
State	*circumstances of an entity*
Transition	*any given time a state change*
Attribute	*a named variable*
Value	*intangible individual entity*
Event	*individual taking place at some time*
Physical Event	*event initiated by a context entity*

- *R* and *Rel* are relations expressed in Table 2. It provides the associations between ontology concepts.

Table 2: Some Associations of Environment Ontology.

Association	**formation**
has_static_property	AtomicEntity→Attribute⇸Value
has_dynamic_property	Causellingntity→StateMachine
can_issue	AutonomousEntity→PhysiEvent
	Causellingntity→PhysicalEvent
has_state	StateMachine→State
has_in_event	StateMachine→Event
has_transition	StateMachine→Transition
source_from	Transition→State
be_triggered_by	Transition→Event
cause	Transition→PhysicalEvent
cause event	$e \uparrow event$
anonymous cause event	$\varepsilon \uparrow event$
cause event sends value	$e \Uparrow event(attr.val)$
in state	$e : state$
send value	$e \twoheadrightarrow attr.val$
receive value	$e \twoheadleftarrow attr.val$
need value	$e \curvearrowleft e'.attr.val$

- As for interaction associations, we use notation set $\{\uparrow, \downarrow, \Uparrow, \Downarrow, :, \twoheadrightarrow, \twoheadleftarrow, \curvearrowleft\}$ to specify the relations between an entity and its events to represent interactions.

We use the following interaction events dependency to capture the relations among state transitions.

- Task Dependency(TSD): If $PostState(T_1) = PreState(T_2)$, then T_2 task depends on T_1, i.e. $T_2 \textcircled{T} T_1$.
- Selection Dependency(SLD): If $PreState(T_1) = PreState(T_2)$, then T_2 selection depends on T_1, i.e.$T_2 \textcircled{S} T_1$.
- Outer Task Dependncy(OTSD): Let $e_1 \uparrow event_1$ represents $entity_1$ causes $event_1$, $e_2 \downarrow event_2$ represents $entity_2$ accepts $event_2$. If $e_2 \downarrow event_2$ should happen after $e_1 \uparrow event_1$, then $e_1 \uparrow event_1$ outer task depends on $e_2 \downarrow event_2$. i.e. $e_1 \uparrow event_1 \textcircled{O} e_2 \downarrow event_2$.

2.2 Environment Ontology of OSD Domain

Domain Entity is environment element in a specific domain, it is depicted by five elements. $DE :=< Ename, TypeSet, StaticPropertySet, DynamicStateMachine, Interaction >$

- *Ename* is entity's name.
- *TypeSet* is the set of Types the entity belongs to. Usually, an entity belong to more than one type.
- *StaticPropertySet* is the set of static property the entity has.
- *DynamicStateMachine* depicts the dynamic properties and state transitions triggered by interactions.
- *Interaction* describes an entity's actions it causes to happen or actions caused by other entities.

Figure 1(Jin and Liu, 2006) shows the environment of On-Line Selling Domain. Each entity is expressed in a rectangle with 5 elements introduced above. Table 3 gives some OTSD of On-Line Selling.

Table 3: OTSD of On-Line Selling Domain.

$Customer \Uparrow pickout(Product.Name.Value)$ $\rightsquigarrow Product \Downarrow BeingPickedOut(Product.Name.Value)$
$MoneyCollector \uparrow HavePaid \rightsquigarrow$ $Customer \downarrow HavePaid$
$MoneyCollector \uparrow HavePaid \rightsquigarrow$ $Product \downarrow ItemsBeenPaid$
$Customer \uparrow Confirm \rightsquigarrow$ $Product \downarrow OrderBeenConfirmed$
$Customer \uparrow Confirm \rightsquigarrow$ $Order \Downarrow Confirmed(TotalPrice.value)$

3 WEB SERVICE DESCRIPTION BASED ON ENVIRONMENT ONTOLOGY

After describing the environment ontology of OSD domain, we present a triplet to define capability description: $CapabilityDes :=< EnvironmentEntityS, InteractionS, Scenario >$

- *EnvironmentEntityS*: *EnvironmentEntity* is the set of environment entities of OSD domain which are involved during the Web Service execution. $EnvironmentEntityS := \{entity | entity \in$

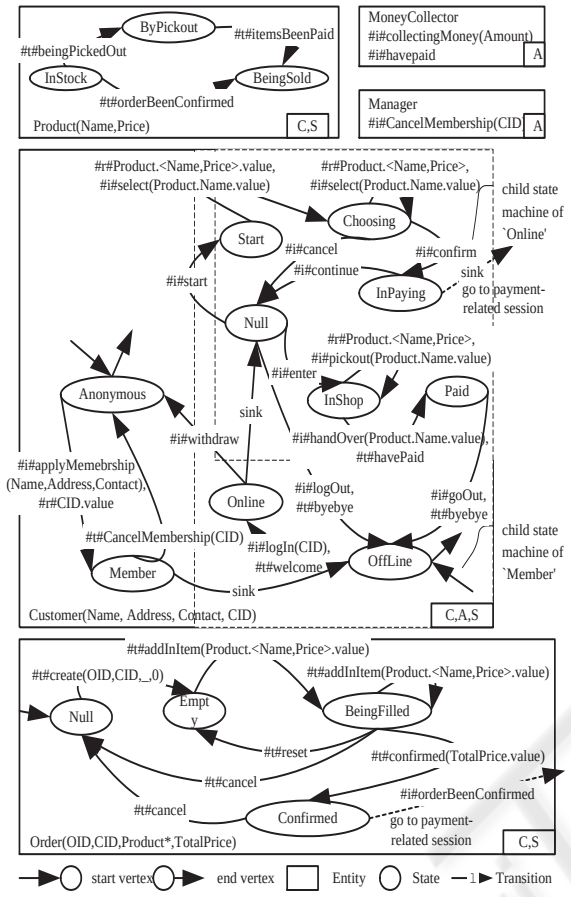

Figure 1: Environment of On-Line Selling Domain.

$OSDEntity\}$,in which $OSDEntity = \{Product, MoneyCollecter, Manager, Customer, Order\}$.

- *InteractionS*: *InteractionS* contains the environment entitys possible interactions. $InteractionS := \bigcup_{Entity_i \in ContextEntityS} Entity_i * EI$.
- Scenario: A scenario element depicts a integral interaction process which may cause several state transitions represented by Effect. $Scenario := \{InteractionSequence || Effect\}$. *IntercationSequence* is the set of sequential Event interactions, which belongs to several environment entities. And $Effect := \{CE : CE.State_1 \rightarrow CE.State_2\}$.

4 WEB SERVICE DISCOVERY

We propose that Web Service request annotation mechanism is the same with Web Service description profile. Now suppose we have a request for a Web Service with this effect:

$$Product : InStock \rightarrow BeingSold$$

The Web Service discovery process works as follows.

Step 1. Capture the request into Web Service description profile.

① Extract the state transition caused by the required Web Service. Find the path which can proceed from start state to end state in context entity's DynamicStateMachine in the guide of Web Service requester. Get the sequence of event interactions along this path: $< INT_1, INT_2, \cdots, INT_n >$.

② Check the dependency of the sequential set elements one by one to get the entire interaction sequence. The algorithm of finding all OTSD and TSD dependencies of an *INT* works recursively as follows:

```
Procedure LocatInteract(INT)
    If INT prefix='#i#'
      Then find post interaction of INT
      INTopo in OTSD;
      Insert it after INT;
      LocatInteract(INTopo);
      Find pre and post interaction of INT
      INTtpr,INTtpo in TSD;
      Insert them  before and after INT;
      LocatInteract(INTtpr);
      LocatInteract(INTtpo);
      End Then
    End If
    Find Pre interaction of INT INTopr in OTSD;
    Insert it before INT;
    LocatInteract(INTopr);
    Find pre and post Interaction of INT
    INTtpr,INTtpo in TSD;
    Insert them before and after INT;
    LocatInteract(INTtpr);
    LocatInteract(INTtpo);
End Procedure
```

③ Construct the request description. Get the related entities when refer to OTSD in step1.2. In step 1.2 it is possible that more than one sequential sets are approved by the requester, which with the effect, the state transition, construct the scenarios. When all these finished, the Web Service request has the form of:

$$< EnvironmentEntityS_{re}, InteractionS_{re}, Scenario_{re} >$$

And the scenario part of request description is showed in table 4.

Step 2. Retrieve the Web Service

① Consider the standby Web Services in Web Service registration center. Every Web Service has the description of:

$$< EnvironmentEntityS_{sd}, InteractionS_{sd}, Scenario_{sd} >$$

Select from Standby Web Service where $EnvironmentEntityS_{re} \subset EnvironmentEntityS_{sd}$ And $InteractionS_{re} \subset InteractionS_{sd}$ And $Scenario_{re} \subset Scenario_{sd}$

With this process, the initial request is for the option "a selling service by order", and the proper Web Service can be retrieved.

Table 4: Profile of Selling by Order.

OrderSeller
Scenario:{
{Customer(CID)↑start,
Order⇓create(OID,CID,∅,0),
Customer↶Product.<Name,Price>.vlaue*,
{Customer⇑select(Product.Name.value),
Order⇓addInItem(Product.<Name,Price>.value)}*,
Customer↑confirm,
Order⇓confirm(TotalPrice.value),
Customer↑continue}
||Product.Name:InStock⇝Product.Name:
BeingSold,
{Customer(CID)↑start,
Order⇓create(OID,CID,∅,0),
Customer↶Product.<Name,Price>.vlaue*,
{Customer⇑select(Product.Name.value),
Order⇓addInItem(Product.<Name,Price>.value)}*,
Customer↑cancel},
||Null,
...}

5 RELATED WORK AND CONCLUSION

WSDL uses syntax ports, OWL-S tries simple semantic pre and post interface of a Web Service process, both method are not semantic enough for automatic feature of Web Services. Different from above work, we construct a domain environment ontology to support Web Service semantic description. Web Service's capability is grounded onto the state transitions of environment entities, and the event interactions among the entities. Based on the environment ontology, we design a matching process for automatic Web Service discovery and selection in Web Service description framework. Also, this ontology-based description framework support dynamic composition of Web Services by combining interaction sequence and scenarios.

However, many issues still need consideration in future. We are trying to use process algebra to describe the interaction of Web Services, representing the capability of Web Services more potently.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Fund for Distinguished Young Scholars of China under Grant No. 60625204, the Key Project of National Natural Science Foundation of China under Grant No.60496324, the National 973 Fundamental Research and Development Program of China under Grant No.2002CB312004, the National 863 High-tech Project of China, the Knowledge Innovation Program of the Chinese Academy of Sciences and MADIS.

REFERENCES

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). *The Semantic Web*. Scientific American, America.

de Vergara, J. E. L., Villagr, V. A., and Berrocal, J. (2005). Application of owl-s to define management interfaces based on web services. *Management of Multimedia Networks and Services*, 3754/2005:242–253.

E.Christensen, F.Curbera, G.Meredith, and S.Weerawarana (2001). Web Services Description Language (WSDL) 1.1. Technical Report NOTE-wsdl-20010315, W3C. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

Fu, X., Bultan, T., and Su, J. (2004). Analysis of interacting bpel web services. In *Proceeding of 13th World Wide Web Conference*, pages 621–630, New York, USA.

Gutierrez, C., Hurtado, C., and Vaisman, A. (2005). *The Semantic Web: Research and Applications*, volume 3532/2005. Springer Berlin Heidelberg.

Jin, Z. and Liu, L. (2006). Web service retrieval, an approach based on context ontology. In *Proceedings of the 30th Annual International Computer Software and Applications Conference*, pages 513–520, Chicago, USA.

M.Jackson (1997). The meaning of requiremens. *Annals of Software Engineering*, 3(5):5–21.

M.Jackson (2001). *Problem Frames: Analyzing and structuring software development problems*. Addison-Welsley.

Motta, E., Domingue, J., Cabral, L., and Gaspari, M. (2003). Irscii: A framework and infrastructure for semantic web services. *Proceedings of 2nd International Semantic Web Conference*, 2870/2003:306–318.

Paolucci, M. and Sycara, K. (2003). Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41.