An Efficient Web Service Annotation Method Utilizing Concept Mapping and Interface Expansion

Guobing Zou[1], Yang Xiang[1], Yanglan Gan[1], Hongyu Sun[1], Zengbao Liu[2]
[1] *Department of Computer Science and Technology, Tongji University, Shanghai, 201804, China*
[2] *Dongtan Coal Mine, Yanzhou Coal Mine Co.,Ltd, Zoucheng, 273500, China*
guobing278@sina.com

Abstract

With the number of services published on the Internet growing at an explosive speed, it is difficult for service requesters to discover satisfactory web services. The reason for this phenomenon is that traditional service organization mode lacks semantic information metadata, which results in low discovery effect. In this paper, we firstly give a service description model and then present an overall framework for service semantic annotation. Based on constructed domain ontology, mapping function of interface concept set and service interface expansion algorithm are proposed respectively. Finally, web services annotation algorithm is presented. Extensible experiment results demonstrate that annotated web services by our proposed method can more satisfy requirements of service requesters than keyword-based described web services. It can achieve higher service discovery effectiveness.

1. Introduction

The internet is emerging not only as an infrastructure for data, but also for a wide variety of information resources, which are increasing being made available as web services [1]. With the rapid development of web services technology in these years, although XML based standards (i.e., UDDI, WSDL and SOAP) has been very mature in registration and discovery mechanism, it is very difficult for requesters' to discover optimal web services. The reason is that current standards focus on operational and syntactic level in the implementation and execution of web services, which limits service discovery process to the keyword-based techniques. Therefore, we should seek for more efficient approaches to implement interoperation and discover user satisfactory web services.

Our investigation has shown that semantic metadata annotation technique can help us solve disadvantages of keyword-based service matching by adopting ontology, which is a formal, explicit specification of a shared conceptualization [2], and can provides domain knowledge. Therefore, we will utilize and semantic context of domain ontology to annotate existing web services and turn them into semantically described web services.

In summary, our main contributions are listed in the following four aspects. Firstly, rather than establishing a dictionary or knowledge index, domain ontology has been designed and constructed in this paper, which is used for mapping function of interface concept set and service interface expansion. Secondly, on the basis of constructed domain ontology, we propose an overall web service semantic annotation framework consisting of five closely correlative components. Thirdly, In order to enrich ample semantic information for web service interface, we provide a mapping mechanism between source service interface and ontology concept. An interface expansion algorithm is also given for expanding mapped service interface set. Finally, we present a service annotation algorithm, which is in charge of annotating web services based on the results of interface expansion algorithm.

The remainder of this paper is organized as follows. Section 2 reviews related work about service discovery and annotation. Domain ontology is defined and modeled in Section 3. In Section 4, we firstly give a web service semantic description model, and then propose the overall service semantic annotation framework, service interface expansion algorithm and service annotation algorithm respectively. Simulation experiment is shown in Section 5. Finally, Section 6 concludes the paper and future work.

2. Related work

Service annotation is the first step but very critical to achieve full scope of web service interoperation, service discovery and service composition. The goal of our work is to give a service annotation method for automatically understanding service function interfaces. In this section, we discuss some related efforts that describe how to add semantics to web services. We also look into some service description models because they are the foundation of our approach to efficiently organize semantic web services.

The work proposed in the literature [1] presented a semantic annotation framework called MWSAF. Authors utilized domain ontologies to categorize web services into domains and implemented semi-automatically marking up web services descriptions. In [3], the authors explored a variety of machine learning techniques, including Bayesian learning and inference algorithm, to semi-automatically

978-0-7695-3735-1/09 $25.00 © 2009 IEEE
DOI 10.1109/FSKD.2009.297

create semantic metadata for describing semantics of web services. In [4], the authors proposed a semantic annotation method by workflow definitions. From the investigation to these service annotation methods, we can conclude that current methods are short of the ability of automatically understanding semantics of service function interface.

There are two main semantic description languages for web services are DAML-S in [5] and OWL-S in [6]. However, modeling web services by these two description languages are really complicated and difficult for service providers to publish their service information and service requesters to submit their service requirements. Therefore, a simple and useful semantic description model is required to help us organize web services.

In order to address these issues, this paper proposes an efficient service annotation method. Our final goal is to automatically understand and annotate web services so that it can improve service interoperability and discovery effect between human and computer.

3. Domain ontology

On the basis of the definition about domain ontology in [2], we respectively give a formal description of domain ontology (*DO*) and semantic relation set as follows.

Definition 3.1 **Domain ontology.** *DO* is defined in a specific domain as a five tuple:

$$DO=\{C, P^C, R, I, A\} \quad (1)$$

- C represents all domain concepts;
- P^C includes all properties attached to C;
- R denotes all kinds of relations among concepts, properties and instances;
- I consists of all instances that belong to concepts;
- A is composed of all axioms in the domain.

In terms of the relation characteristic among concepts, properties and instances, four different kinds of semantic relations are considered in this paper as it appears in the following definition.

Definition 3.2 **Semantic relation set.** In a domain ontology *DO*, *R* is formally denoted as a four tuple:

$$R=\{compose\text{-}of, kind\text{-}of, attribute\text{-}of, instance\text{-}of\} \quad (2)$$

Where, *compose-of* depicts whole and part relation between two concepts. *kind-of* represents the successive relation between two concepts. *attribute-of* describes the relation between concept and its corresponding properties. *instance-of* denotes the relation between a concept and its subordinate instances.

According to above two definitions, we utilize node represent a concept, property or instance. Meanwhile, joint edge is used by four different kinds of semantic relations. Therefore, a directed hierarchy tree (*DHT*) is formed as the structure of domain ontology. A part of the *DHT* for city traffic domain ontology is shown in Figure 1.

From the *DHT* in Figure 1, we can see that '*subway*' is an ontology concept, '*line name*' and '*speed*' are the properties affiliated to '*subway*', '*no.1 subway*' is one of the instances correlative to '*shanghai subway*'.

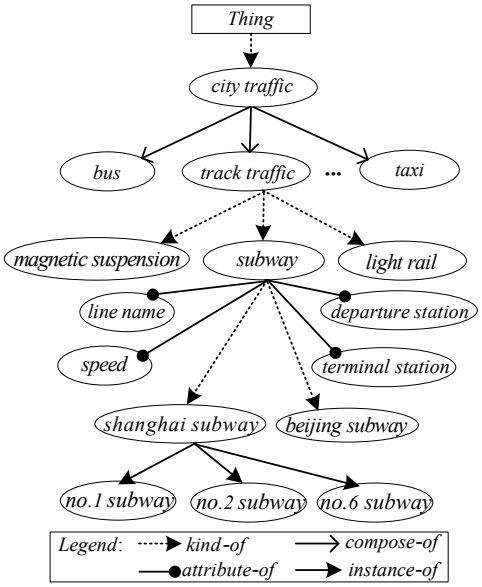

Figure 1. *DHT* segment of city traffic domain ontology

4. Web service annotation method

4.1 Semantic description model

Service description model plays an important role in the process of web service discovery. In order to make web service understood and interoperated between human and computer, it is necessary to model web service for adding semantics to its input/output interfaces. In this paper, we give a light semantic description model *WS-SDM*, which includes web service model and service operation model.

Definition 4.1 **Web service model.** A web service in semantic web service repository is defined as a four tuple:

$$ws=\{wsId, wsName, wsDesp, OprSet\} \quad (3)$$

Where, *wsId* is the unique identifier. *wsName* is the web service name. *wsDesp* is service functional description. *OprSet* is the service operation set, which is formalized as $OprSet=\{opr_1, opr_2, ..., opr_s\}$.

OprSet consists of a series of correlative web service operations. Especially, each $opr_i(1\le i\le s)$ can be executed for a special function. From the analysis of process model in OWL-S, we can conclude that an output interface maps to one or multiple input interfaces. Considering parameter binding relationship in a service operation, we give its definition in the following one.

Definition 4.2 **Service operation model.** A service operation in the *OprSet* can be formalized as a four tuple:

$$opr=\{oprName, InSet, OutSet, IOMap\} \quad (4)$$

- *oprName* is the name of service operation;
- $InSet=\{inP_1, inP_2, ..., inP_m\}$ denotes input interface set, which has m parameters and each inP_i (i=1,2,…,m) is an input interface;
- $OutSet=\{outP_1, outP_2, ..., outP_n\}$ is output interface set, which contains n parameters and each $outP_j$ (j=1,2,…,n) represents an output interface.

- *IOMap* is an I/O mapping function between input and output parameters denoted as $IOMap: outP_j \infty InSet'$, where $InSet' \subseteq Inset$. For an $outP_j$ (j=1,2,…,n), there exists a corresponding subset of *InSet* by I/O mapping.

WS-SDM will be used in the following web service semantic annotation framework.

4.2 Service annotation framework

We propose the general framework of web service semantic annotation called *WS-SAF* as it appears in Figure 2, which is composed of an interface extractor, concept set mapper, interface semantic expander, domain ontology, service annotator, and semantic web service repository.

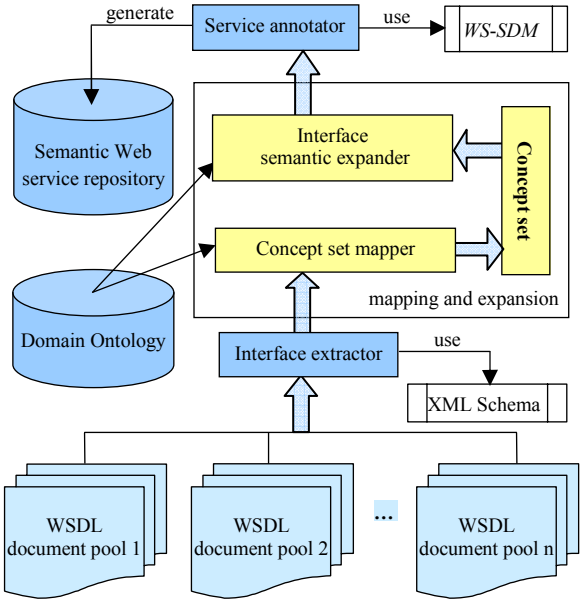

Figure 2. The general framework of service semantic annotation

Interface extractor is in charge of acquiring service file described by WSDL description language from WSDL document pool and then extracting initial input and output interface parameter set with the help of XML schema syntactic structure. The function of concept set mapper is to find corresponding optimal concept set from domain ontology so that it can express the basic interface function semantics for extracted interface parameter set. In order to enrich semantic information of the mapped concept set, we use interface semantic expander to further expand service interface set by semantic analysis and reasoning.

The function of service annotator is to firstly create an empty instance on the basis of *WS-SDM*, and then fill its fields with the semantic information of function interface generated by interface semantic expander. Semantic web service repository stores all the annotated web services that will be discovered and invoked in semantic level. Domain ontology is in the format of an OWL document providing semantic context for mapping and expansion module.

4.3 Interface concept mapping

For a web service, input and output interface in each operation set cannot provide explicit semantic information. Therefore, it restricts the whole process of discovering web services in keyword-based level. Here, a specific mapping function is given to map optimal domain ontology concept that can best match each service interface.

Definition 4.3 **Interface concept mapping**. In domain ontology $DO=\{C, P^C, R, I, A\}$, for a random service interface element $inE \in \{InSet \vee OutSet\}$, there exists an ontology concept *oc* mapped to *inE* from the concept set *C*.

$$oc \leftarrow conMapFunc(inE) \tag{5}$$

In equation (5), concept mapping function calculates similarity between ontology concept and interface element. It includes linguistic similarity and structural similarity. The measure of linguistic similarity value borrowed from match algorithm *NGram* in [7] collects statistics of the common qgrams between name of ontology concept *oc* and interface element *inE*. In structure level, similarity is calculated in [8] between interface element *inE* and all the adjacent subclass concepts of *oc*.

For each interface element $inE_k(1 \le k \le u)$, there is a corresponding ontology concept in concept set *C* that has the maximum similarity with inE_k. Therefore, interface concept set $ICS=\{si_1, si_2, \ldots, si_u\}$ is generated by mapping set function based on single interface concept mapping.

$$ICS \leftarrow mapSetFunc(interface\ set) \tag{6}$$

Where, each $si_k(1 \le k \le u)$ in *ICS* is an ontology concept corresponding to the semantic information for interface element $inE_k(1 \le k \le u)$. *ICS* is used in the following section.

4.4 Service interface expansion

For a concept $c \in C$, its semantic context in domain ontology is yielded by directed hierarchy tree (*DHT*) and semantic reasoning. In order to facilitate description of the following service interface expansion algorithm, several related definitions are firstly given as follows.

Definition 4.4 **Synonym set.** $DO=\{C, P^C, R, I, A\}$, $C=\{c_1, c_2, ..., c_v\}$, for $\forall c \in C$, its synonym set is denoted as:

$$\mathrm{Syn}(c, DO)=\{c_i \mid c_i \in C \wedge c_i \cong c\} \tag{7}$$

Where, $c_i (1 \le i \le v)$ is one of the elements in concept set *C* and has the same meaning with concept c. i.e., With regard to concept '*magnetic suspension*', Syn('*magnetic suspension*')={'*magnetic levitation*', '*Maglev*'}.

Definition 4.5 **Property set.** $DO=\{C, P^C, R, I, A\}$, $P^C=\{p_1, p_2, \ldots, p_w\}$, *DHT* is directed hierarchy tree of *DO*. For $\forall c \in C$, its property is denoted as:

$$\mathrm{Prop}(c, DHT)=\{p_j \mid p_j \in P^C \wedge p_j \lhd c\} \tag{8}$$

Where, $p_j (1 \le j \le w)$ is one of the elements in set P^C and also attached to concept c. Taking concept '*subway*' for example in Figure 1, Prop('*subway*', *DHT*)={'*speed*', '*line name*', '*departure station*', '*terminal station*'}.

Definition 4.6 **Instance set.** $DO=\{C, P^C, R, I, A\}$, $I=\{Inst_1, Inst_2, \ldots, Inst_r\}$, *DHT* is the directed hierarchy tree corresponding to *DO*. For a random $c \in C$, its instance set is defined in the following equation.

$$\mathrm{Ins}(c, DHT)=\{inst_k \mid inst_k \in I \wedge inst_k \prec c\} \tag{9}$$

Where, $inst_k (1 \le k \le r)$ is one of the elements in set I and subordinate to concept c. i.e., taking the '*shanghai subway*' as an example, Ins('*shanghai subway*')={'*no.1 subway*', '*no.2 subway*', '*no.6 subway*'}.

Definition 4.7 **Direct subclass set.** $DO=\{C, P^C, R, I, A\}$, DHT is direct hierarchy tree of DO. For $\forall c \in C$, its direct subclass set is denoted as:

$$\mathrm{Dss}(c, DHT)=\{f_l \mid f_l \in C \wedge (KR(f_l, c) \vee CR(c, f_l))\} \quad (10)$$

Where, $f_l (1 \le l \le v)$ is one of the elements in concept set C. $KR(f_l, \mathrm{c})$ denotes that f_l and c satisfy *kind-of* semantic relation. $CR(\mathrm{c}, f_l)$ describes semantic relation *compose-of* between c and f_l. For instance, Dss('*track traffic*', *DHT*)= {'*magnetic suspension*', '*subway*', '*light rail*'}.

Service interface expansion algorithm called *SIE* is shown in the following Algorithm 1.

Algorithm 1: *Service interface expansion (SIE)*
Input: $DO=\{C, P^C, R, I, A\}$, *DHT*, *ICS*=$\{si_1, si_2, \ldots, si_u\}$;
Output: Interface expansion set *IES*;

```
1.  IES ← NULL ;
2.  For each sik ∈ ICS {
3.      IES[k].insert(sik); //append sik to IES
4.      Syns ← Syn(sik, DO);
5.      If ( Syns ≠ NULL ) //judge synonym set
6.          IES[k].insert(Syns);
7.      Dss ← Dss(sik, DHT); //get direct subclass set
8.      If ( Dss ≠ NULL ) {
9.          IES[k].insert(Dss);
10.         Continue; } //finish sik expansion
11.     Inst=Ins(sik, DHT);
12.     If ( Inst ≠ NULL ) {
13.        IES[k].insert(Inst);
14.        Continue; } //finish sik expansion
15.     Pset ← Prop(sik, DHT);
16.     If ( Pset ≠ NULL )
17.         IES[k].insert(Pset); //add property set
18. }
19. Return IES;
```

In Algorithm 1, we loop each interface concept si_k and generate its interface expansion set *IES* based on domain ontology and its corresponding *DHT*. For each si_k ($1 \le k \le u$), we firstly append itself to interface expansion set *IES*, and then its synonym set *Syns* is yielded and attached to *IES* (*line* 3-6). If generated direct subclass set *Dss* is not empty, we append it to *IES* and finish its interface expansion (*line* 7-10). Similarly, if generated instance set *Inst* is not empty, it is also inserted to *IES* and finishes its interface expansion (*line* 11-14). Under the condition of both empty interface expansion result of direct subclass set and instance set, we further get property set *Pset* and expand to *IES* (*line* 15-17). Finally, *IES* is generated and returned.

4.5 Service annotation algorithm

By utilizing ontology to provide the semantic context, service automatic annotation algorithm is given based on interface concept mapping and service interface expansion. The description of service automatic annotation algorithm called *SAA* is shown Algorithm 2 as follows.

Algorithm 2: *Service automatic annotation (SAA)*
Input: *DO*, *DHT*, *ws*={*wsId*, *wsName*, *wsDesp*, *OprSet*} ;
Output: Annotated web service *aws*;

```
1.  aws ← NULL ;
2.  Define anno_opr, mappedInSet, mappedOutSet;
3.  For each opri ∈ OprSet {
4.      anno_opr.oprName ← opri.oprName;
5.      mappedInSet ← mapSetFunc (opri.InSet);
6.      anno_opr.InSet ← SIE(DO, DHT, mappedInSet) ;
7.      mappedOutSet ← mapSetFunc(opri.OutSet);
8.      anno_opr.OutSet ← SIE(DO, DHT, mappedOutSet) ;
9.      generate IOMap to anno_opr;
10.     aws.OprSet.add(anno_opr);
11. }
12. aws.wsId=ws.wsId;
13. aws.wsName=ws.wsName;
14. aws.wsDesp= ws.wsDesp;
15. Return aws;
```

In Algorithm 2, we respectively get service operation opr_i in *ws* and yield its corresponding annotated operation in *aws*. For each opr_i ($1 \le i \le s$), its operation name is firstly acquired and set to annotated operation *anno_opr* (*line* 4). Subsequently, we use *mapSetFunc* to get mapped input set *mappedInSet*, and then *SIE* is invoked to get input interface expansion set and attach it to annotated operation (*line* 5-6). Similarly, output interface expansion set is generated and appended to *anno_opr* (*line* 7-8). Meanwhile, we generate new *IOMap* in *anno_opr* according to original input and output parameter binding relationship. Finally, we append *wsId*, *wsName* and *wsDesp* of *ws* to *aws* (*line* 12-14).

5. Experiment and analysis

In order to validate the effectiveness of the proposed service annotation method, we have established domain ontology in the area of city traffic by ontology editor Protégé 3.3.1, which includes approximate 220 concepts, properties and instances. At the same time, 362 WSDL web services files about city traffic have been collected for our experiment data set. We have designed a web service annotation prototype based on proposed *WS-SAF*.

We utilize service recall S_{recall} and service precision ratio $S_{precision}$ to evaluate service discovery effectiveness. S_{recall} refers to proportion of matched correlative service number S_{mcn} out of total correlative service number S_{tcn} in web service repository. $S_{precision}$ is defined as proportion of matched correlative service number S_{mcn} relative to total matched service number S_{tmn}.

$$S_{recall}=\frac{matched\ correlative\ service\ number\ S_{mcn}}{total\ correlative\ service\ number\ S_{tcn}} \quad (11)$$

$$S_{precision}=\frac{matched\ correlative\ service\ number\ S_{mcn}}{total\ matched\ service\ number\ S_{tmn}} \quad (12)$$

For comparing service discovery effectiveness of our proposed method with other related methods, we have set up three different kinds of service index repositories based on collected WSDL web services. They are respectively JAXR service registry denoted as *Keyword* by using UDDI registration and discovery mechanism, JUDDI extended registry [9] that is denoted as *OWL-S* by utilizing OWL-S description ontology, and semantic web service repository denoted as *WS-SAM* by adopting *WS-SDM*.

We adopt a set of service requests {sr_1, sr_2, sr_3, sr_4, sr_5} in the city traffic domain to calculate evaluation index. Service recall ratio and service precision ratio among three kinds of methods are shown in Figure 3 and 4.

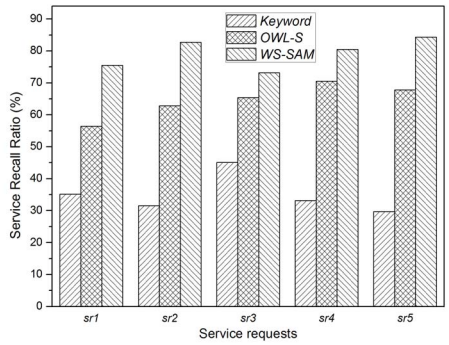

Figure 3. Service recall ratio of three different methods

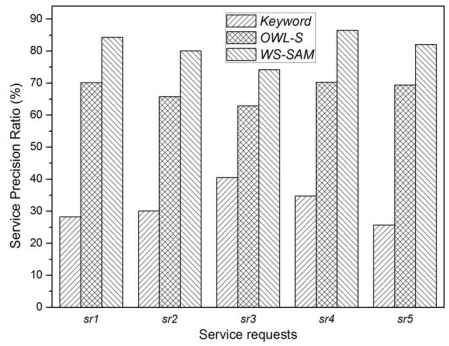

Figure 4. Service precision ratio of three different methods

From experimental results of three different methods, we can conclude that service annotation method proposed in this paper can outperforms other related two methods in a specific service request set. We have constructed domain ontology as semantic context in service annotation process, which can help us understand semantic information of web service function interface and improve service precision ratio. Moreover, we have expanded interface concept set, which can enlarge the scope of web service semantics and improve service recall ratio.

6. Conclusion and future work

By utilizing constructed domain ontology to provide semantic context, this paper has discussed and proposed a new service annotation approach on how to add semantic information to web services. We firstly give a semantic description model *WS-SDM*, and then an overall service annotation framework is presented based on the previous model. Secondly, we extract interface concept of each I/O service interface and generate interface concept set by the given interface concept mapping function and mapping set function. Thirdly, service interface expansion algorithm is presented to generate interface expansion set by expanding interface concept set. Finally, service annotation algorithm is proposed in terms of previous results. So annotated web services contain semantic information and are discovered by matchmaking engine in the semantic level.

There are two major directions in our future work. Firstly, we will mainly concentrate on the improvement of interface concept mapping, and the optimization of more efficient interface expansion algorithm. Secondly, we will further consider how to add semantic weight to annotated web service interface so that it can promote the synthetic effectiveness during the service discovery process.

Acknowledgements

This work is funded by the National '863' High-Tech Research and Development Plan of China under Grant No. 2008AA04Z106, the NSFC under Grant No. 70771077, and the Project of Science and Technology Commission of Shanghai Municipality under Grant No. 08DZ1122300.

References

[1] A Patil, S Oundhakar, A Sheth, et al. METEOR-S web service annotation framework. *In Proc. of the 13th Intl. World Wide Web Conference*, 2004, pp.553-562.

[2] R Studer, V Benjamins, D Fensel. Knowledge engineering, principles and methods. *Data and Knowledge Engineering*, 1998, Vol. 25(12), pp.161-197.

[3] A Heβ, N Kushmerick. Learning to attach semantic metadata to web services. *In Proc. of the second Intl. Semantic Web Conference*, 2003, pp.258-273.

[4] K Belhajjame, S Embury, N Paton, et al. Automatic annotation of web services based on workflow definitions. *ACM Transactions on the Web*, 2008, Vol. 2(2), pp.1-34.

[5] M Burstein, J Hobbs, O Lassila, et al. DAML-S: Web service description for the semantic web. *In Proc. of the First Intl. Semantic Web Conference*. Sardinia: Springer-Verlag, 2002, pp.348-363.

[6] The OWL Services Coalition. OWL-S: semantic markup for web services [EB/OL]. http://www.daml.org/services /owl-s/1.0/owl-s.html, 2004.

[7] E Zamora, J Pollock, et al. The use of trigram analysis for spelling error detection. *Information Processing and Management*, 1981, Vol. 17(6), pp.305-316.

[8] Y Li, Z Bandar, D McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 2003, Vol. 15(4), pp.871-882.

[9] N Srinivasan, M Paolucci, K Sycara. An efficient algorithm for OWL-S based semantic search in UDDI. *In Proc. of the First Intl. Workshop on Semantic Web Services and Web Process Composition*, 2004, pp.96-110.