This is the published version


Available from Deakin Research Online

http://hdl.handle.net/10536/DRO/DU:30005004

Reproduced with the kind permission of the copyright owner

Copyright: 2003, International Society for Computers and Their Applications (ISCA)
Meaningful UDDI Web Services Description

Atul Sajjanhar, Hongen Lu
School of Information Technology
Deakin University
221 Burwood Highway
Burwood, VIC 3125
Australia
{atuls, helu}@deakin.edu.au

Abstract: There is a lack of meaningful description of Web Services in UDDI, however, it is necessary for automatic services discovery. In this paper, a language is proposed for semantics description of Web Services which complements the UDDI standards to derive relationships among Web Services. The semantics description can be used to automatically derive relationships among Web Services. We describe how the semantics description of Web Services, based on the proposed language, can be integrated with UDDI.

Keywords: Semantics, UDDI, Web Services, XML

1. Introduction

UDDI (Universal Description Discovery and Integration) is an industry initiative enabling businesses to define their Web Services, discover other Web Services, and share information about how they interact in a global registry.

To find a specific service in UDDI, a user needs to input basic information about the required service, such as service name, keywords, and then browse the results. Users need to go through hundreds of entries to find the services they are looking for. This process is time consuming and in most cases frustrating.

There have been a number of efforts to add semantics to the discovery process. An early work in this area has been the creation of DAML-S [1], which uses a DAML+OIL based ontology for describing Web Services. The latest draft release [2] of DAML-S uses WSDL in conjunction with DAML-S for Web Service descriptions. Recently, Sivashanmugam et al. [3] have highlighted the complexity and the non-standard approach of implementing DAML-S and they have proposed another approach to add semantics to Web Services.

In this paper, we propose an XML based language to describe the semantics of Web Services. The document based on the proposed XML language will be integrated with UDDI. This description language can be used to automate the process of deriving relationships between services.

We describe the proposed language for semantics description of Web Services in Section 2. The taxonomies in UDDI are described in Section 3. In Section 4, we show how the proposed model applies to a real Web Service. We provide the conclusion and directions for future work in Section 5.

2. Knowledge in Web Services Description

In this section, we describe a language for the semantics description of Web Services. The syntax of the proposed language is based on factors required to represent knowledge in Web Services description. We identified the following factors to describe the semantics of Web Services [4][5].

i. Inputs: This part specifies the objects that a Web Service takes as inputs.

ii. Outputs: This part of the representation specifies the objects that will be the outputs generated by this service.

iii. Input Constraints: This part defines the constraints that expected to hold before this capability can be applied, i.e. the preconditions of this capability.

iv. Output Constraints: This defines the postconditions after the capability has been performed.

v. Input-Output Constraints: This part defines the conditions that should hold across the input and output.

vi. Privacy: Privacy should also be considered, since some service providers or consumers may not want their identities to be revealed to others, whom they know nothing or little about.

vii. Quality: Quality is always a concern of service consumers. Different service providers might provide the same service, but the qualities of their services may vary a lot. By specifying the quality of the required service, consumers have the choice of selecting the service according to their requirement.
The factors identified above are represented in an XML language which has the structure shown below.

```xml
<acdl> ::= <sdlws>
  <condition-language>
    <name>
      <input>
        <value term=""> <name> </value> +
      </input>
      <preconditions>
        <expression> <condition> </expression> +
      </preconditions>
      <output>
        <value term=""> <name> </value> +
      </output>
      <postconditions>
        <expression> <condition> </expression> +
      </postconditions>
      <quality>
        <value term=""> <name> </value> +
      </quality>
      <privacy>
        <value term=""> <name> </value> +
      </privacy>
    </condition-language>
  </sdlws>
  <name> ::= <identifier>
  <condition>::= «expression in condition-language»
```

The root element is "sdlws", which represents the "semantics description language for Web Services". Ontologies form the basis for shared conceptualisation of a domain [6]. The above language uses terms defined in a domain ontology, which is specified in the tModel described in Section 3. The language is independent of any condition language, which enables developers to choose any suitable language to describe the constraints for the Web Services.

3. Taxonomies in UDDI

The UDDI registry describes the Web Services in an XML document. The high level structure of a UDDI entry is shown in Fig. 1. UDDI information model in the Fig. 1 contains four core elements [7].

i. Business Entity Element: This element includes information such as address, contact, and known identifiers. It uses the white pages taxonomies.

ii. Business Service Element: This element includes information such as industrial categorizations such as Industry: NAICS (Industry codes - US Govt.), Product/Services: UN/SPSC (ECMA), Location. It uses the yellow pages taxonomies.

iii. Binding Template Element: It includes technical information about services by referencing to tModel elements (described below). It uses the green pages taxonomies.

iv. tModel Element: This is the technical model element. It is an abstract representation of the technical specifications. It has URL pointers to interface specifications for the service, which can be the WSDL document.

![Fig. 1: High Level Structure of a UDDI Entry](image)
The attributes of the tModel element are below [7].

i. tModelKey: Unique identifier for the technical model
ii. operator: UDDI registry site that manages the master copy of the technical model
iii. authorizedName: Individual who published the technical model

The elements of the tModel element are below [7].

i. name: Readable names recorded for the technical model
ii. description: Short technical model description
iii. overviewDoc: Metadata that describes overview information about the technical model
iv. identifierBag: keyedReference elements that are identifiers for the technical model
v. categoryBag: keyedReference elements that provide categorization information for the technical model

We propose to register a tModel based on a domain specific ontology. After a tModel is registered it can be used for publishing other Web Services in the same domain.

4. Implementation

In this section, we define the SDLWS for a Web Service which provides financial services. The ontology in the finance domain is illustrated in Fig. 2.

![Financial Domain Ontology](image)

**Fig. 2: Financial Domain Ontology**

We use the ontology in Fig. 2 to define the SDLWS for a Web Service that can provide the latest stock price of a Nasdaq listed company.

The tModel element containing meta-data about the ontology for the financial Web Service is shown below. This description is in accordance with the UDDI taxonomies described in Section 3.
In this example, the condition language is specified as FOPL (First Order Predicate Language). The input of this Web Service is a variable type of CompanyName (this term is defined in the finance domain ontology), and it is listed on the NASDAQ stock market. The output is the latest price of for stock of the company. The output from service has a delay of 20 minutes, which defines its quality.

This example illustrates how Web Services can be described based on a domain ontology and the SDLWS language. The methodology is easy to use and it is flexible.

5. Conclusion and Future Work

In this paper, we have defined the syntax for semantics description of Web Services. We have also shown the method for integrating the proposed semantics description with UDDI. An example has been provided to illustrate the technique.

In the future, we will develop agents to find matching services based on the proposed semantics description. Agents can access the UDDI registries using UDDI API. The UDDI API is divided into two components – the inquiry API and the publisher API. Both the inquiry API and the publisher API take the form an XML message placed within a SOAP message envelope [9].

References