This is the published version


Available from Deakin Research Online

http://hdl.handle.net/10536/DRO/DU:30005393

Reproduced with the kind permission of the copyright owner

Copyright: 2004, International Society for Computers and Their Applications (ISCA)
Agents for Invocation of Web Services

Atul Sajjanhar\textsuperscript{1} and Hongen Lu\textsuperscript{2}

\textsuperscript{1}School of Information Technology
Deakin University
Burwood, VIC 3125
Australia
atuls@deakin.edu.au

\textsuperscript{2}Department of Computer Science and Computer Engineering
La Trobe University
Bundoora, VIC 3086
Australia
helu@cs.latrobe.edu.au

Abstract: Web services are becoming popular and widely accepted on the Internet. UDDI is the standard for publishing and discovery of web services. In this paper, we investigate semantics description of web services based on domain ontology; based on this language, we propose an architecture for invoking agents to consume services within the UDDI registry. The semantics service description language together with agent creation architecture provides a new way to discover and utilise published web services. This method is flexible and extendable to accomplish complex web service requests.

Keywords: Intelligent Agents, Semantics, UDDI, Web Services

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.

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 eb 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. Previously, we proposed an XML based language to describe the semantics of web services [4].

web services Definition Language (WSDL) is an emerging communication standard [10] used by web services. Management of the interaction between the broker and the service provider based on WSDL is difficult for non-trivial tasks [11]. We propose an architecture for creation of agents by the broker. These agents will interact with the service provider. We use Agent Communication Languages (ACL) for the communication between agents.

We give an overview about adding semantics to web services. The semantics are integrated with UDDI during publishing. We show how agents can be written, based on the semantics description of web services which is retrieved from UDDI during discovery. These agents are used to invoke methods exposed by web services. The agents can be used to automatically retrieve and collate information supplied by web services. We illustrate the proposed technique by example.

An overview of the proposed system is provided in Section 2. The proposed language for semantics description of web services is described in Section 3. In Section 4, we show the implementation of the proposed model. The conclusion and directions for future work are presented in Section 5.

2. System Overview

The overview of the proposed system is shown in Figure 1. The web services are published in UDDI by the service providers. We complement the UDDI entry with semantics description. The semantics description of web services is described in Section 3. The UDDI publisher API is used to publish the web services. The broker has two functions: discovery of web services from UDDI registry and creation of agents to interact with web services. The broker uses the UDDI inquiry API to obtain information about relevant web services. In the proposed technique, the broker will obtain
semantics description of web services in addition to the basic information supplied by UDDI.

After the discovery process, the broker is able to create agents for invocation of web services. The broker models the agent according to the semantics description gathered during the discovery process.

In Section 4, we describe how the broker is able to create agents based on the semantics information available from the UDDI. The agents created by the broker are used to invoke methods that are exposed by the web services. This allows the broker to obtain and collate the information available from different web services in a specific domain.

![Figure 1: Overview of the System Architecture](image-url)

3. Semantics Description of Web Services

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 [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. 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. The language is independent of any condition language, which enables developers to choose any suitable language to describe the constraints for web services.
The UDDI registry describes the web services in an XML document. The high level structure of a UDDI entry is shown in Figure 2. UDDI information model in the Figure 2 contains four core elements [7][9].

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.

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.

Figure 2: High Level Structure of a UDDI Entry
4. Implementation

In section 4.1, we describe how to publish and discover a Web Service based on SDLWS. In Section 4.2, we describe how agents can be written to automate the process of gathering information from web services.

4.1 Publishing and Discovery

In this section, we define the SDLWS for a Web Service which provides financial services. The ontology in the finance domain is illustrated in Figure 3.

We use the ontology in Figure 3 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.

![Figure 3: Finance Domain Ontology](image)

```xml
<tModel authorizedName="..." operator="...

<name>
  Finance Service
</name>
<description xml:lang="en">
  Ontology for finance services
</description>
<overviewDoc>
  <overviewURL>
    http://ori.cm.deakin.edu.au/FinanceService/_
    ontology.xml
  </overviewURL>
</overviewDoc>
<categoryBag>
  <keyedReference tModelKey="uddi-org:..."
      keyName="Stock market services" keyValue="xmlSpec"/>
</categoryBag>
</tModel>
```

When new tModels are registered, they are automatically given tModelKeys, which are guaranteed to be a Universally Unique Identifier (UUID) [8]. The overviewURL tag of the tModel points to the ontology for the domain. In this case, the overviewURL points to the ontology for the finance domain which is illustrated in Figure 3. The specification for the ontology is described in an XML document, hence the keyValue attribute in the categoryBag is xmlSpec. A Web Service using this tModel will need to reference it in the bindingTemplate.

The sdlws for the finance domain is shown below. This can be obtained by invoking the GetSDLWS method of the Web Service.
In this example, the condition language is specified as FOPL (First Order Predicate Language). The communication language supported by the Web Service is KQML. 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 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 domain ontology and the SDLWS language. The methodology is easy to use and it is flexible.

4.2 Invoking Agents

In this section, we describe how agents can be written to interact with service providers. Consider the financial Web Service described in Section 3. To provide this service, we need to forward this web service description to a corresponding agent, this can be done by wrapping the web service description in the content field of KQML as following:

```xml
<invoke
  :receiver agent1
  :language SDLWS
  :ontology FINANCE
  :content (<sdlws>
    <condition-language>
      FOPL
    </condition-language>
    <input>
      <value term="CompanyName"> ?company </value>
    </input>
    <preconditions>
      ( Listed ?company Nasdaq )
    </preconditions>
    <output>
      <value term="StockPrice"> ?price </value>
    </output>
    <postconditions>
      (Equals DateOf(?price) Today)
    </postconditions>
    <quality>
      <value term="delay">20 minutes</value>
    </quality>
  </sdlws>)
```
The receiver agent is able to interpret the KQML message, and invoke a process to deal with this web service. Since all the conditions are written in first order logic, it is simple to automatically generate a process, or an agent, that can accomplish the required task.

5. Conclusion

In this paper, we give an overview of a language used to describe the semantics of web services. We show how the proposed semantics description can be integrated with UDDI. The semantics description is used during the discovery process to create agents. The automatically created agents are used to invoke methods on web services. We show how this technique can be used in retrieval and collation of relevant information from domain specific web services.

References


