An Adaptive Metadata Model for Domain-Specific Service Registry

Kun Chen¹,², Yanbo Han¹, Dongju Yang¹, Yongshan Wei¹,², and Wubin Li¹

¹ Research Center for Grid and Service Computing Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
² Department of Computer Science and Technology, Shandong University of Science and Technology, Qingdao, China
{chenkun, yhan, yangdongju, weiyongshan, liwubin}@software.ict.ac.cn

Abstract. A domain-specific service registry should satisfy two requirements at least: coping with diverse service description requirements from different services; supporting semantic description on service interfaces for interoperation in a specific domain. This paper proposes an adaptive metadata model that supports flexible semantic description of service interfaces. It uses a simple inheritance mechanism to provide multiple metadata models on a light-weight generic metadata model and semantic annotation templates to facilitate interface semantic description for domain-specific services. The implementation and application of this metadata model in a real-world domain-specific service registry promises that a user can customize a metadata model and add interface semantic metadata in an easy-to-use way.

1 Introduction

As the basic function unit of SOA, services have a most commonly agreed-upon aspect that they are encapsulated reusable business functions defined by interfaces [4]. Consequentially, how service consumers discover appropriate services turns to be a key problem in SOA. Service registry acts as information mediation between service providers and service consumers. Since service description information is usually stored as service metadata, service registry puts metadata model a basic position. As a result, requirements on metadata model should be well considered when designing a service registry.

In fact, we are just facing some fair and reasonable requirements in a real-word project. The project is called Agriculture Technology Information Integration System (ATIIS), which uses Web services to integrate distributed agriculture technology information stored in different locations and provides a uniform access point for ender users. Agriculture Technology Information Service Registry (ATISR) is a part of it, which could register various services provided by different organizations and map their interface schema into an integrated view with domain ontology.

⋆ This work is supported by the National Natural Science Foundation of China (Grant No. 70673098) and the Project of “Taishan Scholar” which is Funded by Government of Shandong Province.
—Multiple Metadata models can be defined. Different services have different requirements on service description. In order to be adapted to the diversity of requirements, the definition of multiple metadata models should be supported.

—Interface semantic annotation should be supported. WSDL is a type of syntactic description. However, service discovery and interoperation need to know the meaning of service interface parameters. Domain ontology is used to add semantic annotations on WSDL interface parameters to tackle this problem.

The project is a representative in the applications of using Web services to integrate distributed and heterogeneous data. The same requirements are also existed in a wide spectrum of other scenarios such as e-government system and healthy care management, in which Web services are used for providing information and domain ontology is introduced to add interface semantic metadata.

A large amount of service registries have been proposed by academic world as well as some IT vendors [14]. However, most metadata models in them try to describe all aspects of a service, which makes them complex and not easy to use. In this paper, motivated by the above-stated requirements, we propose an adaptive metadata model which uses a simple inheritance mechanism to provide multiple metadata models for various services. Moreover, annotation template is used to facilitate interface semantic annotation for domain-specific services.

2 Related Works

As an industrial standard, UDDI [3] aims at providing a universal registry data model. It uses TModels to provide additional data to the UDDI core entities for different service descriptions. However, the taxonomic structure of TModel is complex and using it for extension requires extending the discovery interface, which is almost impossible for ender users. Moreover, UDDI core entities do not support semantic metadata. OWL-S [6] and WSMO [5] are two major initial and most popular semantic Web service description models. Although they both have extension mechanisms, we can not get a special extended model for the description of a particular group of services. WebSphere Service Registry and Repository (WSRR) developed by IBM [4] uses templates to model sets of metadata, allowing many different metadata models to be represented. However, for template definition, WSRR currently exploits only the subTypeOf relationship in OWL, which provides no better ability than the taxonomy used in TModel.

Much work has been done on semantic annotation. MWSAF [2] uses a media structure called SchemaGraph to facilitate the matching between XML schema and ontology to find the relevant ontology. Unfortunately, the matching results are not good if the WSDL file does not have a good structure or the ontology becomes much comprehensive. [8] proposes a template-based markup tool for web contents semantic annotation. But because web contents are various, structure and reusability of templates are main problems.
3 The Adaptive Metadata Model

3.1 Multiple Metadata Models

Two kinds of metadata models are related by the inheritance mechanism:

**Generic metadata model**: Generic metadata model is used to organize the common metadata items among all services.

**Extended metadata model**: Extended metadata model can be either inherited from the generic metadata model or another extended metadata model. It is used by a particular type of services for description.

General metadata model should benefit for four activities for a service: discovery, substitution, composition and management [10]. Here, four types of general metadata in it are summed up. We think they can describe a service throughout its entire lifecycle, and provide necessary information for service application and service management.

- **Technical metadata**, including service URL address, operations, input/output information and so forth, give the information of where the service is and how to use it.

- **Semantic metadata**, such as operation semantic metadata, input/output semantic metadata, enable exact service discovery and service composition.

- **Profile metadata**, which can be unstructured text including service description, organization name and so on. These metadata can be used by service consumers as well as the administrator of a registry for service browsing.

- **Management metadata**, including service using times, publishing status and other records throughout a service's lifecycle. These metadata facilitate the administrator to manage registered services and service consumers to discover appropriate services.

All the profile metadata are put in a ServiceProfile property, management metadata in a ManagementInfo property, and technical metadata and semantic metadata in a ServiceGrounding property. The generic metadata model includes the three properties for describing common properties of all services.

Extended metadata are used to define special metadata items for a particular type of services. They can be referred by an extended metadata model. We define an extended metadata item as an ontology class in OWL, so it can describe various concepts a user needs and support semantic discovery.

In order to make the definition of extended metadata models easy, we consult to the inheritance mechanism in object-oriented filed, which allows for the definition of a subclass that inherits the features of a specific superclass [9]. Inheritance mechanism is a good method for reuse and specialization. Here, we only permit the simplest single inheritance. That means a sub metadata model can inherit only one super metadata model with its own extended metadata items.

This saves work, because users only need concentrate on new features. For example, a metadata model F for describing acquiring information services has been defined, and this metadata model has two extended metadata items:
Subject and Contents. Now, if a user wants to describe payment acquiring information services, he can just define a new metadata model \( N \) by inheriting \( F \) and adding a new extended metadata item: \( \text{Price} \). The user needs not care metadata in \( F \), for \( N \) has inherited all of them automatically (Fig.1).

3.2 Semantic Annotation Templates

Services are encapsulated reusable business functions defined by interfaces. Business functions represent certain activities in a domain, and domain activity knowledge has commonness and is reusable [7]. Based on such a fact, we use an interface semantic annotation template to describe a business function. An interface semantic annotation template consists of four parts:

- **Template Name:** describing business function of a Web service operation.
- **Operation:** describing the operation with domain ontology.
- **Inputs:** describing input parameters of the operation with domain ontology.
- **Outputs:** describing output parameters of the operation with domain ontology.

Semantic annotation templates can be established by domain experts beforehand. A service may have several operations. However, in order to facilitate business function description, a semantic annotation template describes exactly one operation. So a service has to be split into several function units when using interface semantic annotation templates.

A template describes service interface schema with domain concepts and can be reused. On the one hand, semantic annotation templates hide ontological complexity from users, which liberates them from the burden of manual relating concepts in Web services to a domain specific ontology. On the other hand, as semantic annotation templates encapsulate and expose the domain knowledge and service interface schema to a service provider for configuration, the semantic annotation accuracy is higher than that of automatic semantic annotation methods which try to relate ontology with no meaning WSDL interface names.
4 Implementation and Application

ATISR is a part of ATIIS. It builds on the metadata model proposed in this paper. ATISR is implemented by three major components: Metadata model manager is used for managing multiple metadata models for diverse requirements on service description; Interface semantic annotation templates manager is used for mapping service interface schema into an integrated view; Service metadata manager provides CRUD operations on service metadata for the application layer in ATIIS such as task analyzer and results integrator, service monitor and service execution and interoperation engine.

To better explain how the registry works, a real application scenario is described as follows: A user uses the registry to publish a payment Web service that searches the expert information stored in Beijing Agriculture Bureau. He can use ATISR to achieve his goals with the following major steps:

Choosing or defining a metadata model: This user can browse all the metadata models have been established in the registry. He finds his service can be described by the searching information service metadata model, but he wants to declare the cost of his service. So he defines Price as an extended metadata item, and created a payment searching information service metadata model which inherited the searching information service metadata model with the extended metadata item: Price.

Filling in metadata items in the metadata model: After choosing the payment searching information service metadata model, ATISR will initial some metadata in this metadata model automatically, such as management metadata. The user should fill in or choose other metadata through a GUI.

Choosing and configuring interface semantic annotation templates: Then he chooses the searching expert information template and configures the sequence of the input and output parameters in the template according to his service WSDL file.

Finally, if the user submits, ATISR will check if these metadata consist with the payment searching information service metadata model, and stores them into a semantic database.

5 Conclusion

In order to satisfy the diversity of requirements on service description and support service interface semantic annotation in an easy-to-use way, we present an adaptive metadata model. The registry metadata model is implemented in ATISR. A real scenario described in 4 validates that a user can customize a metadata model he needs, and add interface semantic metadata easily and accurately. The metadata model only support simple singe inheritance now, however, we believe that more complex inheritance such as multiple inheritances will enhance its extensibility. Moreover, semantic annotation templates have strict
restrictions on service interfaces now, how to make templates flexible is also among our future research goals.

References