Agent-based Mediation in Semantic Web Service Framework

Renato de Freitas Bulcão Neto¹, Yathiraj Bhat Udupi², and Steve Battle³

¹ University of São Paulo, São Carlos SP 13560-970, Brazil, rbulcao@icmc.usp.br
² North Carolina State University, Raleigh NC 27695, USA, ybudupi@csc.ncsu.edu
³ Hewlett Packard Laboratories, Bristol BS34 8QZ, UK, steve.battle@hp.com

Abstract. In a semantic web service scenario clients and services should inter-operate by allowing a service to be delivered via different protocols and data formats. This paper describes a novel solution to protocol and data mediation through a goal-driven, agent-mediated interaction with web services described by OWL-S ontologies. Our contributions include: (i) an OWL-S compiler which mediates between two OWL-S service description interfaces and outputs a script containing a set of executable NUIN plans, and (ii) an agent-based mediator built upon NUIN framework that executes these plans in a event-driven fashion.

1 Introduction

There is a need for richer knowledge-based product and service descriptions to enhance the existing business interactions over the Internet. The present model depends on web services being deployed in a purely syntactic approach. This paper describes an approach to enrich the current web services model using semantic models, which supports for a goal-driven, agent-mediated interaction with web services described by OWL-S ontologies [1]. We present a case study about a software licensing marketplace for vendors, who lack a comprehensive sales infrastructure. A service request made using SOAP-based interactions with the web service enables the client to place an order. Presently hard-coded Java applets are used to help the client to communicate with the web service. For a business client with a browser, it simplifies to have a web friendly, resource-oriented service interaction [2]. The client requests can be simply encoded within the request URL in a resource-based manner, unlike sending explicit request messages. This poses a mediation problem, where we need to enable clients and services to interoperate by allowing a service to be delivered via different protocols and data formats. There are two perspectives of the mediation problem, first is protocol mediation: how we describe one service in terms of another and ensure that it achieves the same goals. The second is data mediation: how we achieve independence from the syntax of the specific messages allowing us to map from one message format to another.

Our approach is set out in the web service modelling framework (WSMF) [3] and it caters to the main objectives of WSMF. It supports rich, declarative service descriptions, which separates the design of the service functionality from its delivery and provides for a framework in which those descriptions are used. Mediation is achieved within an
agent-based framework moving from the syntactic domain of messages into a representational framework based on semantic web technologies. This agent-based mediator assists the client in achieving specific goals, which are seen as the key to identify the tasks and actions to be performed by the service.

**Contribution.** Our first contribution is the development of an OWL-S compiler which mediates between two different OWL-S service descriptions derived from the requestor (the client), and the service provider interface and outputs an executable NUIN script of plans [4]. NUIN is a framework with emphasis on the building of Semantic Web agents. Our second contribution is the development of the agent-based mediator built upon NUIN framework that executes the script generated from the compiler in an event-driven fashion. These two provide a novel solution to the mediation problem and aid in e-business process automation.

## 2 Agent-Based Mediation

This section describes the whole process of mediation being enabled by the agent-based mediator. The event-driven invocation of plans is also demonstrated with the help of an example scenario.

### 2.1 The Agent Framework in the BDI Architecture

The agent framework which mainly animates the mediation and the WSMF can be seen in terms of a belief-desire-intention (BDI) architecture [4]. The framework maps the various elements of the conceptual architecture into agent beliefs, desires and intentions. Beliefs correspond to the background knowledge of the agent held in its knowledge base (updated with message content at run-time) and accompanying ontology. Desires include information about the client goals, comprising of the information in the service request which is derived from the OWL-S profile through service parameters. The intent of the client is conveyed to the agent through individual requests at the user interface. This way the agent translates the desires and intents from the requestor interface into tasks and actions at the provider interface.

NUIN provides for the BDI architecture to build the agent. The agent executes the various NUIN plans that perform the required protocol and data mediation. These plans coordinate activities across the various plug-in components that support communication with the client and the service provider. The Figure 1(a) describes the whole agent architecture. The web plug-in of the agent mediator functions as an adapter between a web server and the agent, transforming messages back and forth into HTML. The service plug-in of the agent acts as an adapter to an invocation client for the SOAP web services. A lift module provides an interpretation of the message content and a translation to or from a common representational form, an RDF model, with the support of a mapping based on XML schema [5]. A request message gets dropped from RDF to XML and similarly the response is lifted to RDF to allow for data mediation.

### 2.2 Protocol Mediation by Process Planning with the OWL-S Compiler

Process planning is an off-line process that generates the NUIN plans required to mediate between the requestor and the provider interfaces, and is performed by the OWL-S
compiler. Both interfaces include OWL-S service descriptions of the semantics such as inputs, outputs, preconditions, unconditional effects, service parameters, etc. The service descriptions at both the interfaces need not often have a one-to-one mapping between the processes they describe. Where the immediate effects of actions at the two interfaces do not correspond exactly, we can define composite processes that have the required effect. Also, a business abstract process model represents the abstract view of the provider interface. This ontological model is imported by the concrete processes of the two interfaces. The primitive parts of the abstract process are of type OWL-S SimpleProcess allowing us to describe a process independently of its realization.

The OWL-S compiler takes in the above descriptions and outputs a set of executable NUIN plans. The compiled output is modular in such a way that each plan corresponds to an atomic process. Every composite process is again compiled to produce NUIN plans for their corresponding component processes. This modular design supports backtracking to evaluate the preconditions of the processes, so that we can back out of a plan where preconditions do not hold. These NUIN plans define how the agent bridges between the different protocols of the two service interfaces (protocol mediation).

2.3 Data Mediation using Mapping Rules

The agent-based mediator equipped with a Rules plug-in performs the data mediation, which realizes the mapping between the incoming and outgoing message content and their common ontological conceptualizations. Mapping rules are applied to the content stored in the knowledge base previously received and lifted message content. The rules plug-in is based on the Jena rules engine and hence the mapping rules are expressed in the Jena rules language [6].
2.4 Event-driven Intent Invocation

The agent plans are designed to allow for event-driven triggering of plans. At the requestor interface, the web plug-in translates the request URL from the web server into a form, from which we can extract the request query and the query parameters. The plug-in later embeds these parameters in an event and raises this event to make the agent aware that the request has arrived. Thus an event triggers a particular plan in the agent which performs the required operation of process mediation and the corresponding intent gets transformed into an invocation call at the provider interface. Plans can also invoke other plans and hence the response from the provider interface is conveyed back to the user interface by returning it back through the executed plans.

2.5 An Example Scenario

The Figure 1(b) describes a scenario of the invocation of an atomic process at the requestor-side interface being translated to a composite process with its component atomic processes in the provider-side interface. The step (1) is an event that alerts the agent to the user intention to add an item to the order. The step (2) initiates the composite process, which eventually initiates the individual atomic processes addLineToOrder and getOrderSummary in steps (3) and (4) respectively. Given the intention to addToOrder we use the OWL-S CompositeProcess in a top-down way, while the user intent is signalled to the agent in a bottom-up way through the messages received. Also, the atomic process at the requestor-side and the composite process at the provider-side are connected through the OWL-S SimpleProcess.

3 Conclusion

This paper described an agent-based solution to protocol and data mediation following the major objectives set out by the WSMF: services, mediation, ontology, and goals. We have shown how OWL-S service descriptions may be used within an agent-based framework to support this ontology-based mediation.

References