Enabling Agent-Based Management of Web Services with WS2JADE

Xuan Thang Nguyen, Ryszard Kowalczyk
Faculty of Information and Communication Technologies
Swinburne University of Technology, Melbourne VIC 3122, Australia.
{xnguyen, rkowalczyk}@swin.edu.au

Abstract

Web services management has attracted substantial research and industry interests in recent years. Web services technology is gaining popularity because of its well defined infrastructure aiming at enabling interoperability among heterogeneous applications. However, this interoperability promise also poses a difficulty in building a Web service management framework which can work across organizational boundaries. Software agents have been recognized as a promising technology for managing Web services. However in many proposed frameworks no agent standard is assumed. In this paper, we propose an extension of our existing Web services-agent integration toolkit WS2JADE for Web services management with FIPA compliant multi-agent systems. In particular, we relate our framework to other solutions in the area and show how new emerging Web services management technologies can be used with WS2JADE for enabling Web services management with agents.

1 Introduction

With the emergence of Web service standards, the universal interoperability between distributed applications is fast becoming a reality. Web services follow a loosely coupled integration model and use industry-standard protocols to facilitate the seamless integration of heterogeneous systems within and across organisations. While the Web service technology offers many distinctive advantages and benefits, it still has certain limitations potentially hindering its broader adoption in more complex applications. In particular it involves the limited support for the management of the discovery, composition and execution of Web services (e.g. [1, 12, 19]).

The agent technology offers abilities of intelligent operations, interactions and cooperation between autonomous components that can be used in automating management tasks and business processes, and has also been recognized as a promising technology for managing Web services [10, 11, 20]. However, because Web services and agents were originally developed separately with different standards and specifications their integration is not straightforward. Realising the benefits of integrating these two technologies, significant research has been carried out in this direction. The Agentcities Web Service Working Group’s project [2] is an example of such an effort aiming at addressing the issue of Web services and agents integration. More recently, Agentcities has also created openNet [22] that provides a test-bed environment for integrating software agents, Web services and Semantic Web services. The integration of software agents and services in general has been proposed by Luck et al. [14] as one of the major tasks for the agent community. The main obstacles in integrating Web services and agents are the mismatches in description and communication used by these two technologies. We had implemented a toolkit, WS2JADE [3], to overcome these obstacles. WS2JADE follows a proxy-based integration approach that allows the two technologies to evolve in parallel without imposing any restrictions on either, providing the gateway to bridge the Web services and agents.

Web services can work across organizations and be composed to create new Web services. There are business scenarios in which many Web services from different administration domains need to be tied together in cross-organisational business processes or complex composite applications. Consequently, Web services management becomes a complex task that requires a high level of intelligent capabilities and automation support. The emerging Web services management technologies, such as WS-Management [1] and Web Services Distributed Management (WSDM-MUWS [18], WSDM-MOWS [19]), define additional interfaces for Web services needed for their management. However they do not specify the management mechanisms, i.e. ‘how’, ‘when’ or ‘why’ these Web services can be managed. The agent technology is well poised to fulfil this role and support Web services with the required management mechanisms. If agents were able to interact with Web services, the agent technology could be integrated into the Web service management model effectively and used to manage Web services.

In this paper, we propose an extension of our WS2JADE toolkit for seamless and dynamic management of Web services through agent technology. To do this, it utilizes useful features from emerging Web service management models; especially Web services based management models like WSDM and WS-Management. It pro-
vides facilities to integrate with future implementation of WSDM and WS-Management. It is the first step towards the ultimate goal of automation of Web service management using agents. Our WS2JADE extension does not automatically provide intelligent algorithms for Web services management. However, it provides a gateway for the existing management algorithms to be employed in the multi-agents environment. With WS2JADE extension, an overlay management network of agents can be formed to manage the Web services network and communicate with the underlying management of Web services.

The paper is organised as follows: It starts with an overview of the related work in Section 2. Section 3 presents the WS2JADE extension for Web service management with FIPA agents. Examples of agent-based management of Web services with WS2JADE are presented in Section 4. Concluding remarks and an outline of the future work are presented in Section 5.

2 Related work

Agents to Web services communications as well as the synthesis of the agents and Web services have been addressed in a number of works. The trend of using agents to monitor and control Web services composition has been increasing recently, evidenced by a number of other publications (e.g. [7, 12, 20]). In most of those works however, agents do not conform to any specific standard and details of using agents to invoke Web services are not formally described. The general assumption is that any agent can request any Web service by acting as the Web service’s client. In practice, if this assumption held then the agents’ code would need to contain the Web service invocation code. For FIPA-compliant agent systems, this also means that in addition to agent communication languages, the agents’ programmer need to consider the low level details of Web services invocation. There have been attempts to provide a framework in which agents and Web services, with separation of concerns in their implementations, can communicate with each other. A concrete implementation towards this direction is WS2JADE [3]. WS2JADE provides facilities to deploy and control Web services as agent services at run time for deployment flexibility and active service discovery.

The area of Web services management has attracted substantial effort from industry and significant research among academic communities. Web services-based applications can work across enterprise boundaries more now than any other types of applications. However, being distributed and dynamic in nature, Web services require an efficient management model that can integrate seamlessly and work dynamically in a distributed environment. There have been two main approaches in tackling this problem and they are complimentary to each other. In the first approach, management is done through Service Level Agreement (SLA) or service contracts. This approach assumes that control over Web services is not visible to external managers and service management is enforced through agreement terms (rewards, penalty, preference, etc) specified in the contracts. Current work on this approach can be found in [9] and [21]. In the second approach, external managers can exert direct control over a Web service. For this to happen, the Web service that needs to be managed must expose some manageability interfaces. This approach is the latest emerging management model from industry, supported for example by two specifications, Web Services Distributed Management (WSDM) [16] originally from HP and WS-Management [1] from Microsoft. At this initial stage, WS-Management and WSDM appear to overlap in a number of aspects.

Web services represent vital resources for any business organization and are prevalently used to carry out business processes and transactions between businesses or within an enterprise. Hewlett-Packard has proposed the Web Services Management Framework (WSMF) which is a logical architecture for the management of resources [16]. Extending this work further, with the support from other companies including IBM and DELL, HP has released the Web Services Distributed Management (WSDM) specification which defines how the management of any resource can be accessed via web service protocols – Management Using Web Services (MUWS) [18] and how Web services can be managed using Web services – Management of Web Services (MOWS) [19]. The specification was submitted to Organization for the Advancement of Structured Information Standards (OASIS) and has been accepted as a standard. Microsoft, in collaboration with various IT companies has released its own SOAP-based protocol for managing systems (including Web services) called Web Services for Management (WS-Management) [1]. WS-Management shares the basic idea with WSDM in identifying a manageable resource and communicating with it.

A management system, in general, requires some level of automation. In an ideal situation, it should be able to seamlessly diagnose faults and take appropriate correction actions. Therefore a management system itself or its sub-components shares the fundamental characteristic of an agent, and some attempts have been made to use agents for Web service management systems. Examples are Sun’s Java Management Extension (JMX)-a new feature in version 5.0 of Java 2 platform and McAfee’s ePolicy Orchestrator [15] software. In JMX, a given resource is instrumented by Java objects known as Managed Beans, or MBeans. These MBeans are registered and managed by management agents, known as JMX agents. The specification provides a set of services for JMX agents to manage MBeans. In ePolicy Orchestrator, Anti-Virus software on client PCs are monitored by a set of agents for possible virus outbreaks. In these examples, the management environments in which the managed objects (MBeans, Anti-Virus software) reside do not scale up globally. In particular, the environment is limited to Java programming lan-
guages for JMX and to a local computer for ePolicy Orchestrator. Consequently, the management agents in these local environments do not need to follow well-defined standards. However, Web services environment can span organizational boundaries, and hence, its management agents, if implemented, should exhibit social capabilities with a common standard such as FIPA specifications so that coordination can take place at a global level.

3 Enabling Web Service Management with WS2JADE

This part discusses the application of agent technology for Web service management and WS2JADE extension’s capability in integrating with emerging Web services based management models. We begin with a review on WS2JADE existing architecture and explain how JADE agents to access and use Web service. This is an important pre-requisite for any Web service management framework with agents. We then discuss how WS2JADE extension can be used to manage Web services.

3.1 WS2JADE – Access and use Web services

WS2JADE allows JADE agents to access Web services through its Web service proxy agents called WSAG. WSAG are capable of not only invoking Web services but also translating Web services’ data structure and interaction models into agents’ ones. All of these translations/mappings are done dynamically at runtime. These translations/mappings involve the ontology mapping (translation from Web service WSDL into agent ontology), interaction mapping (translation from Web service choreography to agent protocols), and assignment mapping. These mappings are handled by three main components that form the WS2JADE: ontology generation and management component, interaction translation component, and the service assignment management component. These components interact with each other and generate ontologies as well as dynamic WSAG on the fly.

Figure 1 presents the different components within WS2JADE system and how they are linked to JADE agent platform. The vertical rectangular box depicts WS2JADE, the horizontal one depicts JADE. Note that the overlap between WS2JADE and JADE consists of components in the WS2JADE interconnecting layer: generated interaction protocols, ontologies, and WSAG. Figure 1 also illustrates a scenario for WS2JADE operation, in which a client agent searches for some service on DF. The DF can trigger WS2JADE to look up for available services in the Web service environment. If some Web services are found, their corresponding ontology and interaction models are generated. Also, a WSAG capable of accessing the Web service is generated. This WSAG registers the Web service as its service on DF, and communication between the client agent and this WSAG can start if the client agent wants the service.

3.2 WS2JADE extension for management

Web services management is a broad term which covers different areas such as access mechanism (authentication, authorization, etc), provisioning (SLA management, execution monitoring, etc), and composition (composition structure, conversation relationships, etc). As mentioned before, the MUWS 1.0 specification defines rules to represent and access manageability interfaces of resources as Web services. The MOWS 1.0 specification represents Web services as resources and describes access mechanisms to the resource manageability interfaces using MUWS. They together provide the Web services managers with one set of protocols and semantic instrumentation to manage Web services based applications and processes across enterprise and organizational boundaries. However, to what extent a manageability interface can be exposed to external managers and which techniques the Web service managers use to coordinate and manage their Web services efficiently and intelligently are not in the scopes of these specifications. In other words, while these Web service management standards define the interfaces needed for managing a Web service, they do not specify ‘how, ‘when’ or ‘why’ a web service should be managed. WS2JADE extension provides a framework for gluing agent intelligent and social capabilities and the Web services management structure together. The agents can coordinate, cooperate and negotiate on a particular level of visibility exposed by a management interface and make plans of how to manage Web services. Figure 2 illustrates this. Direct Web service manipulations are supported by management infrastructures which implement WSDM or WS-Management. In the top layer of WS2JADE and FIPA compliant MAS, available AI techniques can be implemented and distributed among FIPA compliant...
agents, these agents communicate with WS2JADE agents to get access to the manageability interfaces of manageable Web services in the second layer of Web service Management. They then coordinate and manage the Web services in the bottom layer. Note that these are logical layers. In some implementation, services in the Web service management layer can be the same with services in the bottom layer.

Figure 2: WS2JADE for WSMangement

As discussed before, WS-Agreement and WS based management are two major directions in managing Web services. WS2JADE extension improves WS2JADE by adding new JADE ontology bases to describe the agreements and manageable resources. It also provides facilities to translate existing WS-Agreement documents and WSDM manageable interfaces into these ontologies. The extension leverages WS2JADE capabilities in accesses and controls of Web services.

Figure 3 depicts the ontology model used to capture important concepts in a WS Agreement document. The relationships between concepts are “part of” relationship. ContractAgreement defines an agreement between the service provider and the service consumer on how to use a service (Web service). It consists of ServiceContract and InteractionContract. InteractionContract, at this stage of WS2JADE extension, refers to the simple request/response interaction model of Web services. It can be extended in the future to include more complex interaction models, especially for composite management Web services. ServiceContract corresponds to the abstract service in a WSDL interface. A collection of OfferedItem objects forms a particular class of a service. An OfferedItem can be a QoS parameter like response time or availability, or any attribute that can distinguish different services by its values, for examples memory and CPU cycle allocations. Guarantee Terms are conditions on the values of OfferedItem. Different guarantee terms can be grouped together to form a new guarantee term. A Guarantee Term has a string expression in XPATH language. Information on XPATH can be found at http://www.w3.org/TR/xpath. WS2JADE extension allows agents to construct Agreement ontology objects from an input WS Agreement document. This functionality has been implemented in ciamas.wsjade.ws2jade.wsdm.wsa package. A ContractAgreement object can be retrieved as follows:

```java
ContractAgreement agmt = ciamas.wsjade.ws2jade.wsdm.wsa.WSAReader(inputFile);
```

Figure 3: Agreement Ontology Model

We believe a Web based management of Web services model, such as WSDM or WS-Management, has a great potential to be widely adopted by software vendors in the future. Hence, we have implemented an ontology base for Web services management in WS2JADE. The ontology structure was designed after a careful review of both WSDM and WS-Management specifications to combine common and important management elements in these specifications. Figure 4 partially presents this ontology structure. As illustrated from the figure, WSResource and WSRelationship are important concepts for Web service management. They extend from the WSManagement concept which serves as a root concept of all other content elements. WSResource defines a distinct type of management attributes exposed by a manageable Web service. WSRelationship class represents relationships between two Web services or two Web services resources. At the lowest level, three types of relationships are defined: includesOf, dependsOn (between two Web services), and correlatesWith (between two Web services resources). “includesOf” relationship between two Web services means that the first Web service is composite and it has the second Web service in its composition structure. “dependsOn” relationship keeps track of relationships hidden by virtualization process. A “correlatesWith” relationship indicates a correlation between property values of two Web services’ resources. WSManagementAction is the root concept where all other management actions subclass from. These management actions include WSResourceCreate, WSResourceDelete, and WSResourceUpdate which correspond to resource management functions of creating, deleting, and setting a resource in WSDM and WS-Management. Also, for resource enumerations, WSResourceEnumerate establishes the resource enumeration context and WSResourcePull iterates over an enumeration result set. Depending on how much control over a service that a manageability interface has, the Web ser-
vice management ontology can be further extended. FIPA compliant agents and WS2JADE agents use the ontology to communicate on Web service management related information. Whether a direct control of Web service is required, WS2JADE agents translate the control information into management actions available in WSDM or WS-Management. With the help of Web service management ontology and the management model outlined in Figure 4, management reasoning and access for Web service control can be decoupled and handled separately.

Figure 4: WS Management Ontology Model

4 Examples

In this example, we present a demo as a proof-of-concept of how JADE agents can access and control Web services with WS2JADE extension. In our scenario, a PALM device accesses stock quotes information from a financial Web service. If the server which hosts the Web service becomes unavailable, a JADE agent, acts as the WSDM manager, is able to detect the failure and searches for a replacement of the Web service on the Internet. The agent then reconfigures the PALM application to use the new Web service.

Figure 5 explains the scenario in details. The Stock Quote App is a Web service based application, which exposes a manageability interface also as a Web service. This application displays the current stock quotes of requested symbols as shown in figure 5. The application retrieves information of these quotes from another financial Web service on the Internet. The address of the financial Web service’s WSDL interface is a resource (WS-RF term) of the application and hence can be reconfigured by the JADE agent through the manageability interface. The resource has its value of WS1 address at the beginning. WS1 is simulated to be offline and the JADE agent searches for WS2 as a replacement of WS1.

Figure 5: Palm Stock Application

Figure 6: Application on Palm OS simulator

The demo has been implemented with Palm OS emulator and kSOAP [13] on the mobile device. We built a minimal implementation of WSDM for our Stock Quote App with only one resource to be managed and the management endpoint is assumed to known in advance. If the effort on CIM-WSDM mapping [10] is successful, low level resources like memory allocation and CPU cycles can be managed effectively by agents in the management layer.

5 Conclusion

The paper presents an extension of WS2JADE toolkit for managing Web services with JADE agents. In particular, it adds a new management ontology base to WS2JADE and incorporates common Web services management capabilities in WSDM into the toolkit. The future
of intelligent agents with autonomous capabilities, which manage and access the widespread Web services infrastructure, is promising. The discussion shows how the toolkit extension achieves first steps in that direction. Since Web services is a volatile area with rapid changes and many specifications in WS-* domain (such as WS-Agreement Specification, WS-Resource Framework) need to be implemented and tested out for a richer collection of management capabilities, WS2JADE has been designed to accommodate future plug-in components.

However, there are still software features and design areas that require new solutions, implementations or further improvements. In particular, agent-based Web services negotiation, semantic processing, and full integration with WSDM specification are part of our current work. Web services negotiation will give WS2JADE agents the ability to read, understand contracts, and employ negotiation mechanisms to contract Web services and their compositions according to WS-Agreement specification. We are looking forward for a wide adoption of WSDM or WS-Agreement specifications for a global level of WS management with FIPA compliant agent systems. We hope that WS2JADE can be applied in more real-world examples to make contribution into agent-based Web services and business process management in particular, and practical applications of multi-agent systems in general.

Acknowledgements

This work has been partially supported by DEST in the context of the project Adaptive Service Agreement and Process Management (AU-DEST-CG060081).

References