Hybrid Composition of Web Services and Grid Services

R. Anane, K-M Chao
DSM Research Group
School of MIS, Coventry University, UK
{ r.anane, k.chao } @coventry.ac.uk

Y. Li[1]
E-Business Research Center,
Fudan University, China
liys@fudan.edu.cn

Abstract

Web service composition is now seen as a focal point of research, especially as mechanisms for the coordination of distributed tasks are acquiring more importance. Models such as BPEL4WS can cater for local and centralised coordination. There is, however, a need to reconcile different technologies on a wider scale. This requires the development of efficient and flexible frameworks in order to ensure the optimal use and coordination of distributed applications. The work presented in this paper is concerned with the development of a compositional framework, which brings together Grid services, Web services and Semantic Web technology within a BPEL4WS platform. It is aimed at enhancing BPEL4WS by allowing for the hybrid composition of Web services and Grid services, and by incorporating dynamic binding through agent mediation. The efficient management of workflows afforded by BPEL4WS is combined with the versatility of agent technology. An agent-based system, called SOA, was developed to support the framework.

1. Introduction

The integration of distributed applications and their coordination can be achieved either locally by means of process management (mainly XML-based) technologies, or globally by means of agents and Semantic Web technologies. In XML-based approaches, this has seen the development of compositional languages for workflow management, such as BPEL4WS. The underlying models are relatively efficient and lead to predictable behaviour, with interactions taking place in a relatively stable environment. They implement a static binding policy, and operate at the syntactic level. BPEL4WS operates on Web services only.

The emergence of grid computing is part of wider initiative aimed at introducing frameworks that can facilitate seamless interoperation between distributed applications. Grid computing frameworks, such as the Globus Toolkit, have so far been orthogonal to Web-based technologies, which resulted from two main approaches to distributed systems, XML and Semantic Web. The advent of Web services and their potential for increased interoperation has seen a convergence between Grid computing and Web-based technologies. The introduction of the Open Grid Services Architecture (OGSA) is a testimony to the success of Web services as a model for distributed computation, with the Open Grid Services Infrastructure (OGSI) as an interaction model for Grid services.

The integration of distributed applications and their coordination can be achieved either locally by means of process management (mainly XML-based) technologies, or globally by means of agents and Semantic Web technologies. In XML-based approaches, this has seen the development of compositional languages for workflow management, such as BPEL4WS. The underlying models are relatively efficient and lead to predictable behaviour, with interactions taking place in a relatively stable environment. They implement a static binding policy, and operate at the syntactic level. BPEL4WS operates on Web services only.

In addition to support for interoperation Semantic Web technologies promote a higher level of automation than process management technologies. Agent technology, in particular, is called upon to play a crucial role in enhancing the functionality of Web services. In one approach, for example, agents were generated from Web services by using wrappers [1]. The flexibility and pervasiveness afforded by Semantic Web technologies may, however, lead to complex and computationally expensive interactions.

This paper is concerned with the presentation of a compositional framework, which allows for the seamless composition and workflow management of Web services and Grid services. This requires a transformation of Grid services into Web services, and the establishment of a symbiotic relationship between agents and Web services, within a BPEL4WS platform.

The rest of the paper is organised as follows. Section 2 gives an introduction to Web-based technologies and the roles of agents. Section 3 presents the architecture of the proposed framework. Section 4 offers a motivational case study. Section 5 gives a brief evaluation of the framework, and Section 6 concludes the paper.

[1] Corresponding Author

2. Web-based technologies

This section covers the salient features of the software technologies, which are relevant to the proposed framework.

2.1 Web Services Composition

One fundamental characteristic of Web services is that the selection and binding of services can be performed dynamically, thanks to the decoupling between their ownership and their use. The utility of Web services is further enhanced by the introduction of mechanisms for composing them in order to generate new Web services and applications. The composition of Web services is defined as a process that enables the creation of composite services, which can be dynamically discovered, integrated, and executed to meet user requirements.

BPEL4WS is emerging as an industry standard for Web service composition [2]. A new Web service can be generated from the aggregation of other Web services, and its interface can be described as a set of WSDL *PortType*s, in the same manner as for atomic Web services. Closely linked to BPEL4WS is BPWS4J, an engine that takes as input a BPEL4WS script and WSDL definitions of the bindings for the partners, and produces a point of entry for the BPEL4WS process, as a single Web service. BPEL4WS and one of its engines, BPWS4J, offer predicable behaviour and performance. BPEL4WS has, however, some limitations, notably its centralised workflow enactment and the fact that Web services must be known and bound *a priori* [3]. The early binding of services, defined beforehand, may lead to inefficiencies due to the sub-optimal selection of Web services, and to potential service discontinuity.

The main criticism levelled at XML-based technologies, in general, such as BPEL4WS and UDDI, is that they operate at the syntactic level, are implementation focused and require human intervention at various stages [4]. More significantly, however, is the reliance on WSDL, and therefore on XML and XML Schema for describing Web services. The lack of semantics of WSDL restricts the scope of the operations that can be performed on Web services to publication, discovery, invocation and monitoring. In contrast, semantic technologies such as OWL-S can support more automation, autonomy and meaningful interaction between services.

2.2 Semantic Web technologies

The implementation of flexible frameworks for interoperation depends on a successful integration of Web services and on their mediation by agent technology. There is, however, a gap between XML-based constructs and tools such as WSDL, and the concepts manipulated by agents. This semantic gap can be bridged by Semantic Web technologies [5]. For Web services description, the development and introduction of OWL-S is a significant factor in matching service providers and service requestors [6]. The drive towards the introduction of richer semantics has eased the deployments of agents. OWL-S is an ontology, which provides richer Web service description, in terms of objects and complex relationships. An OWL-S ontology has three components:

1. ServiceProfile: describes what the service does, its inputs and outputs and its preconditions and effects (IOPE); this is equivalent to UDDI content in that it supports the automatic discovery of services.
2. ServiceModel: describes how the service works (control and dataflow in its use). This is similar to BPE4WS.
3. ServiceGrounding: describes how the service is implemented and provides a mapping from OWL-S to WSDL. The ServiceGrounding is a point of convergence between Web services and OWL-S.

There is a symbiotic relationship between agent technology and semantic Web technologies, in particular OWL-S [7]. Agents are suitable for highly dynamic environments and operate at a conceptual level. The main characteristics of their behaviour are autonomy, proactiveness, reactivity and social ability. BDI agents [8] are particularly apt at exploiting the semantically rich environment defined by OWL-S ontologies. These agents hold beliefs (B), have desires/goals (D) and use Intentions/plans (I) to achieve their goals. An agent can be generated and instantiated from an OWL-S description by the following mapping:

- The ServiceProfile in OWL-S maps to an agent's beliefs (B).
- The ServiceModel is mapped to a set of intentions associated with plans (I). Each activity in the process is associated with a sub-plan. Preconditions and effects from the ServiceProfile will translate into conditions and effects for the BDI plan. The elements in the ServiceGrounding are used to define a set of actions within plans.
- The desire (D) is specified by additional functionality in conjunction with the specification of the ServiceProfile.

The ServiceProfile and the ServiceModel in OWL-S provide the semantics, while the ServiceGrounding is used to generate the interface signatures.

2.3 Grid Services

The OGSI specification utilises the WSDL and XML schema definition languages from Web services to define an extended component model [9]. The specification seeks to address issues that occur in complex distributed applications, such as the management of distributed long-lived states. In order to achieve this aim, OGSI introduces

the concept of a Grid service instance. "A Grid service instance is a (potentially transient) service that conforms to a set of conventions (expressed as WSDL interfaces, extensions, and behaviours) for such purposes as lifetime management, discovery of characteristics, notification, and so forth." [9]. The OGSI specification inherits the interoperability features from Web services, but includes additional features:

- Stateful interactions: serviceData is the OGSI approach to stateful Web services. It exposes the state data of a service instance to service requestors for queries, updates and change notifications.
- References: OGSI uses Grid Service Handles (GSH) to name and manage Grid service instances. A client wishing to communicate with a service instance must map the GSH to a Grid Service Reference (GSR).
- Collection of service instances: OGSI allows services to be grouped together, with defined relationships, so that clients maintain them easily.
- Life Cycle management: This gives a client the ability to create and destroy a service instance according to its requirements.

The OGSI specification is an attempt at creating a favourable environment for the management of Grid services. There are some fundamental differences between the coordination of Web services and Grid services. Coordination between Web service instances is driven by data requirements in BPEL4WS. The correlation of Web service instances in BPEL4WS is similar to the handling tables of database systems through index keys. Thus, developers have to define correlation sets from *PortTypes* in WSDL and use them to correlate instances. On the other hand, OGSI uses Grid service instance references to coordinate Grid service instances. Each Grid service instance has a unique reference (similar to an object reference). Since the Globus Toolkit 3 (GT3) is mainly implemented through the JAXRPC specification (a Web Service specification based on Java RMI), the management of a collection of instances is similar to handling multiple instances in Java. As a result, GT3 cannot export its Grid service instance references to BPEL4WS, and BPEL4WS cannot hold references to the Grid service instances. BPEL4WS does not support any construct that allows Web service instances to be destroyed. Instead, it supports the termination of the whole process.

3. Proposed framework

The development of a compositional framework raises two main issues related to the execution of distributed applications. Concern over risks of failure, especially for processes of long duration can be addressed by a stateful policy and by a decentralised implementation. Efficiency and optimal use of resources promote the dynamic search for and discovery of appropriate services.

3.1 Architecture

We propose a compositional framework for supporting hybrid composition of Web services and Grid services. This requires the provision of mechanisms for overcoming some of the limitations of BPEL4WS. The first limitation stems from its incompatibility with Grid services. The second limitation, as highlighted above, concerns the requirement for Web services to be known and defined (bound) before they are incorporated as partners. The third limitation relates to the level of semantics and the lack of autonomy in Web services, characteristics that restrict their scope for participation in distributed applications and agent technology. Circumventing these limitations and enhancing BPEL4WS brings to light a number of issues [10]. The first issue is concerned with the generation of Web services and the description of potential partners. The second relates to the means for storing and using the descriptions for discovering and selecting partners. And the third issue arises from the need to incorporate discovered partners into the BPWS4J engine. These issues are dealt with in the proposed framework, by integrating virtual Web services, OWL-S descriptions and agent technology.

The framework defines two levels, one concerned with the specification of compositions, and the other with the enactment of the process. The scope of the first level is determined in the composition process by the introduction of virtual Web services (VWS) for representing potential partners, in conjunction with ordinary, concrete Web services. Potential partners can be either Grid services or Web services, dynamically bound and mediated by agents. Virtual Web services offer a flexible means for decoupling the composition process from the binding of Web services. A virtual Web service specifies its input and output requirements and associates itself with a nominal *PortType***.** It is defined and used in the composition in the same manner as an ordinary Web service, with the main focus on input/output. Composition within this framework involves incorporating concrete Web services when known and statically bound and Virtual Web services when a Grid service is required or when a Web service is unknown and dynamically bound. The second level is concerned with the binding of the VWS, and therefore with the enactment of the BPEL4WS process. This matter points therefore to the need for a mechanism that implements 'late/dynamic' binding. This form of binding can be performed by the VWS itself as a proxy for a Grid service or by an agent, in the case of a Web service.

In addition to the decoupling of Web service instantiation from composition, agent mediation promotes the dynamic search for and discovery of Web services. This feature is, however, predicated on the availability of

high-level semantics that can be provided by OWL-S. An agent can be generated in two steps. A VWS, mainly identified by its input/output (I/O), is first augmented with richer semantics provided by OWL-S; the VWS is now endowed with IOPE properties. The link between VWS, a Web service, and OWL-S is through WSDL. A BDI agent is subsequently generated from the OWL-S description, as indicated in Section 2.2. In the process, the agent obtains the semantic description and combines it with its reasoning mechanism, in order to acquire the ability to filter Web services, through matchmaking. Once the agent is created from the OWL-S structure it acts as a proxy for the corresponding Web service, and the OWL-S description is kept for documentation and reasoning purposes. The resulting agent is similar to a broker, since it takes requests from a VWS, performs the necessary search/discovery, and then invokes the corresponding Web service represented by the VWS.

3.2 Implementation Issues

The focus of the implementation was on the determination of the structure of the VWS in its interaction with the Grid services and agents. In order to facilitate Grid service integration into a BPEL4WS composition, Grid services are wrapped as Virtual Web services, as shown in Figure 1. All the interfaces defined for the Grid services are re-defined in JavaBeans as an XML complex type (in WSDL) with a public Grid service instance attribute. The WSDL code associated with Virtual Web service, in this case, includes a number of name spaces and *Partnerlinks*. The role of the Virtual Web service in WSDL is generated through the combination of a meaningful name and a random value, so that BPEL4WS is aware of its existence, and redefines it as needed in the workflow. An additional operator is defined and implemented in the Virtual Web service in order to create new Grid service instances [11].

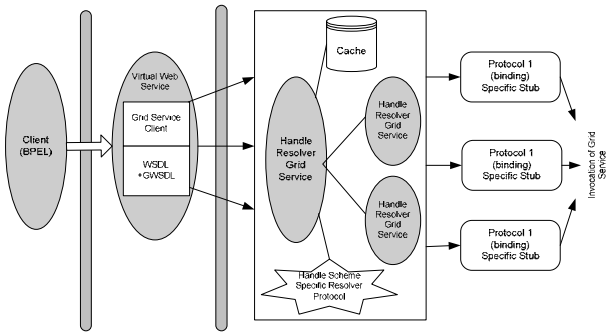

Figure 1. VWS and Grid Service

In agent mediation, a VWS includes operators that trigger events in the agent when it receives the request from the BPWS4J engine, or provides an input to BPWS4J when the agent relays the required input from other agents or Web services. The VWS contains no actual operations, only interfaces to agents.

In order to facilitate the process of creating service-oriented agents, a template agent was designed with two elementary functions: one for listening to events triggered by a VWS and the other one for passing the response from other agents to the VWS. The required input and output must be specified in the agent. The template agent also includes a number of generic coordination protocols such as contract net, English auction, reverse auction etc. A matchmaking mechanism is required for discovering appropriate Web services, according to the syntactic signatures and semantic requirements of the composition process. For this purpose, the auction coordination protocol is used to coordinate the selection process.

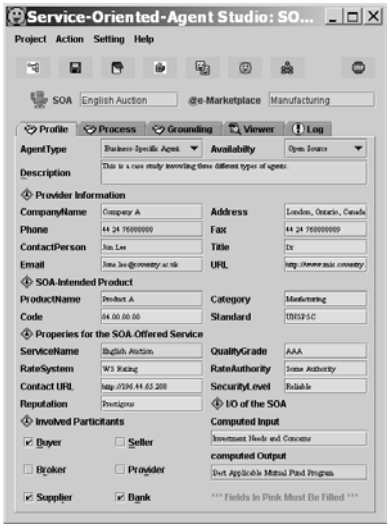

Figure 2 Service-Oriented Agent Studio

We have implemented a system in order to support the architectural framework presented above. The system, called SOA (Service-Oriented Agent) 1.2 studio [12] includes a template for creating VWS and a template for creating agents (see Figure 2). An interface for the system was designed so that users can describe the semantics of Web services and can incorporate OWL-S descriptions. Support for Web services is provided by JAXRPC 1.3 and Tomcat, whereas the IBM BPWS4J 2.1 engine and the BPEL4WS Editor Eclipse plug-in are used for the creation and interpretation of workflows. Agent technology relies on JADE agent for reasoning and communication, and on OWL JessKB [13] for the reasoning capability over OWL-S profiles in the agent. Once the required information is specified, the studio allows the generation and compilation of code to take place in the same environment. The studio offers a seamless combination of design environment and run-time environment.

4. A Case Study for SOA

This section presents a case study, which illustrates the capability of the SOA system. The scenario concerns a manufacturer who wishes to purchase parts for a series of products, and invites a number of suppliers to take part in a bidding process. The manufacturer needs to evaluate the capabilities of the suppliers and their quality of service. The selection of a supplier involves complex negotiations over issues such as price, quantities, and delivery time; the determination of the overall cost follows a number of steps, to which each department in the company is required to contribute. Once the cost is established, the manufacturer determines whether a loan from a bank is required. The banks, in turn, need to assess the manufacturer's credit and reputation in order to approve the loan. These various processes lead to the selection by the manufacturer of one offer from one bank. These processes can be automated by using Web services and agent technologies, within the SOA framework.

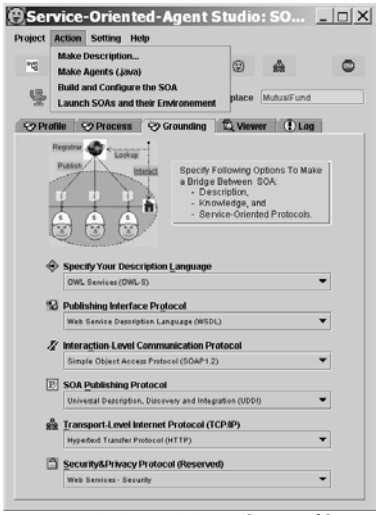

Figure 3. OWL-S Grounding Profile

In the scenario, there are two types of interaction between companies and departments. Each participating company, such as supplier, manufacturer or bank, delegates to an agent the task of supervising the coordination process. Agents use coordination protocols to communicate and discover/select services and reach an agreement to carry out a plan. As the relationship between departments within a company is static, each company has a relatively fixed workflow to model the interactions among its departments. There is, however, a certain degree of non-determinism in the behaviour of the process, but it can be prescribed. BPEL4WS is used to model their activities and ensure consistency. The manufacturer agent needs to communicate with the supplier agent and the bank agent. The manufacturer agent adopts the auction coordination protocol to select one of the suppliers to carry out the negotiation and the transactions based on their agreement. Figures 2 shows a form provided by SOA which enables users to enter the ServiceProfile information in OWL-S. On completing the ServiceGrounding profile, as shown in Figure 3, the user can store it as a project, generate the essential Java code, compile it, and then run the resulting program. SOA creates the agents and deploys them automatically, as shown in Figure 4. Negotiation and the interactions between the agents can take place.

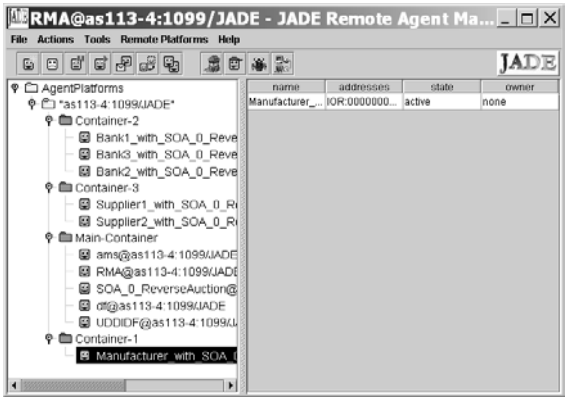

Figure 4. Participating agents

5. Evaluation and Related Work

In this section an attempt is made at putting the proposed framework in context, by considering two other approaches to automation. They are characterised by the role and the level of involvement of semantic technologies in the coordination process. In [3] the rationale is to move away from the rigidity of workflow enactment of BPEL4WS/BPWS4J to the decentralised and flexible mode of coordination of multi-agent systems. The work is presented as one approach among many in Web services composition and enactment, and aims at producing a multi-agent enactment from BPEL4WS composition. This approach has the merit that it offers greater flexibility and can lead to the optimised use of resources. Its main drawback, however, is the added complexity entailed by a transition from one domain of execution to another, and the need to ensure that functional equivalence is achieved and maintained between two different specifications. The fundamental issue is whether the benefits afforded by agent technology, such as flexibility, outweigh the drawbacks of complexity and the increase in computational resources.

Closer to the work presented in this paper is the model described by [10]. BPEL4WS is enhanced by Semantic Web technologies as a means of overcoming the limitations of BPWS4J, and in particular the requirement for *a priori* service definition. BPEL4WS is extended with a Semantic Discovery Service (SDS), which acts as a dynamic proxy between BPWS4J and the potential partners to be located and selected. All requests to

previously selected partners are directed to the SDS, which implements a late binding policy. The SDS is agnostic as to the content of the requests it deals with, and is stateless.

Although our work is less ambitious in scope, the model we propose offers more flexibility and customisation because each Virtual Web service is associated with an agent. The decentralisation of the discovery process makes the system more reliable and more scalable and avoids the single point of failure of the SDS. Furthermore, unlike the SDS an agent can be stateful, learn and be aware of the content of requests it deals with. These features may, however, be costly in computational and in storage terms. The model we propose has also the added advantage that it supports hybrid composition of Web services and Grid services. It may require a heavier human presence in the loop than the SDS-based model, since semantic enrichment is a crucial step in the creation of agents. Both models maintain the original composition, but the SDS model deals only with Web services.

The approach promoted by the framework allows for an incremental development of composition. It exploits the fact that composition in BPEL4WS is seen in terms of processes that interact with *partners* that are *external* to the composition itself and identified only in terms of abstract interfaces. This separates the different concerns through a two-stage refinement process. This approach strikes a balance between the two extremes, one of total enactment by agents, and the other of conformance to the original BPEL4WS static model. The resulting framework offers an enhanced means of combining the predictability of BPEL4WS enactment with the versatility of Semantic Web technologies. From the design point of view the introduction of Virtual Web services, as a decoupling factor, offers the possibility for hybrid composition of Web services and Grid services.

Since the proposed system is to provide an integrated environment for developing agent-based Web services, usability, as a criterion for evaluation, acquires special significance. Users can take advantage of the templates that the system provides for the creation of agents and their coordination. They can easily enhance Web services with semantics through a user friendly GUI. The system combines input from users and templates and generates the necessary code. This can reduce design time and ensure consistency.

6. Conclusion

The architectural framework presented above overcomes BPEL4WS limitations and promotes wider coordination by combining Virtual Web services and agent technology. Although this approach may be computationally expensive, it facilitates the composition of hybrid services and capitalises on the efficiency afforded by the BPEL4WS platform and the capabilities of Semantic Web and agent technologies. This is achieved by wrapping Grid services, and by enhancing Web services through a symbiotic relationship with agent technology. The system is operational, and work is currently being carried out on further integration of the different technologies, and on enhancing agent mediation.

References

[1]. Knoblock C., Minton S., Ambie J.L., Muslea M., Oh J. and Frank M, Mixed-initiative, multi-source information assistants, *Proceedings of the World Wide Web Conference*, ACM press, New York, NY, 2001, 697-707.

[2]. Khalaf R., Mukhi N. and Weerawarana S., Service-oriented Composition in BPEL4WS, *http://www2003.org/cdrom/papers/alternate/P768/choreo_html/p768-khalaf.htm*, 2003.

[3]. Vidal J.M., Buhler P. and Stahl C., Multiagent Systems with Workflows, *IEEE Internet Computing*, January/February 2004, 76-82.

[4]. Richards D., van Spunter S., Brazier F.M.T. and Sabou M., Composing Web Services using and Agent Factory, In *AAMAS Workshop on Web Services and Agent-Based Engineering*, 2003, 57-66.

[5]. Sirin E., Parsia B. and Hendler J., Filtering and selecting Semantic Web Services with Interactive Composition Techniques, *IEEE Intelligent Systems*, July/August 2004, 42-49.

[6]. Richards D., Sabou M., van Splunter S. and Brazier F.M.T., Artificial Intelligence: a Promised Land for Web Services, In *The Proceedings of The 8th Australian and New Zealand Intelligent Information Systems Conference (ANZIIS2003)*, 10-12 December 2003, Macquarie University, Sydney, Australia, 205-210.

[7]. Hendler J., Agents and the Semantic Web, *IEEE Intelligent Systems*, March/April 2001, 30-37.

[8]. Rao, S. A., and Georgeff. M. P., BDI Agents: From Theory to Practice, *Conference Proceedings of 1st international conference on multiple agent system*, 1995, 312-319.

[9]. OGSI, Open Grid Services Infrastructure Version 1.0, *http://www-unix.globus.org/toolkit/ documentation.html*.

[10]. Mandell D.J. and McIlraith S., The Bottom-Up Approach to Web Service Interoperation, *International Semantic Web Conference* 2003: 227-241.

[11]. Kuo-Ming Chao, Muhammad Younas, Nathan Griffiths, Irfan Awan and R. Anane, Analysis of Grid Service Composition with BPEL4WS. *Proceedings of the 18th IEEE International Conference on Advanced Information Networking and Applications (AINA 2004)*, Tokyo, Japan, March 2004, 284-289.

[12]. Yinsheng Li, Hamad Ghenniwa and Weiming Shen**,** Agent-based Web Services Framework and Development Environment, *Journal of Computational Intelligence*, 2004.

[13]. Joseph B. Kopena and William C. Regli, DAMLJessKB: A Tool For Reasoning With The Semantic Web**,** *2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, USA, October 20--23 2003.