52 DYNAMIC ADAPTATION, COMPOSITION AND ORCHESTRATION OF WEB SERVICES IN VIRTUAL ENVIRONMENTS

Peter Bertok
Stephen Reynolds
RMIT University
Melbourne, AUSTRALIA
peter.bertok@rmit.edu.au

To solve interaction and discovery problems, Web services need to be unambiguously described. Existing technologies, such as WSDL describe the functional aspects of Web services, network service end points and interfaces. The semantic aspects are more difficult to handle, as different organizations operate in different ways, and may use different information models and domain specific vocabularies.
This paper explores how ontologies can be used to orchestrate dynamic Web service compositions. A shared ontology is developed that exchanges data and meaning. By adding "smarts" to the service description and not including within the applications' processing of the data, the descriptions are able to move freely between domains. Examples from a bookshop service are used indicate that the proposed method's is well suited to virtual environments.

1. INTRODUCTION

The World Wide Web is shifting from a predominantly information interaction platform operating with HTML documents to a service interaction platform operating with Web services [1], but existing technologies offer only partial solutions [2]. The power of Web services lies in the successful management of the relationships and composition of cooperating business services in a distributed application-to-application (A2A) or business-to-business (B2B) environment. Web service composition is very complex, due to the high autonomy, high distribution, and high heterogeneity of the services involved [3].

Today's Web services are predominantly isolated remote data services. An example is an international online trader that has built an intra-organizational service that operates within a localised environment. Our goal was to build reactive and adaptive Web services that can automatically reason with different business schemas to find a matching service. If a matching service is found, it invokes the service to form a virtual business environment. This happens automatically without user intervention.

At the same time, that most existing Web service composition and orchestration methods are assuming an unknown, but fairly static environment, where once a matching service has been found, invocation and execution is straightforward. The solution proposed here addresses the issue of failures by detecting failed services and invoking replacements.

Web service architecture is implemented via the Web Service Technology Stack (Table 1). Data moves up and down through the layers, each layer addressing a separate business problem.

Table 1: The Web Service Technology Stack [2]

Layer	*Description*
Discovery	Means for consumers to fetch descriptions of providers
Description	Description of the service, contact point and its use
Packaging	Date encoding, serialization and marshalling
Transport	Application-to-application protocols. TCP, HTTP
Network	Addressing and routing

Adaptation and composition of Web services is implemented via the Description and Discovery service layers. In the proposed system, the existing Description and Discovery service layer were extended, to avoid the need to rewrite other parts of the infrastructure because of the specialization of the layers.

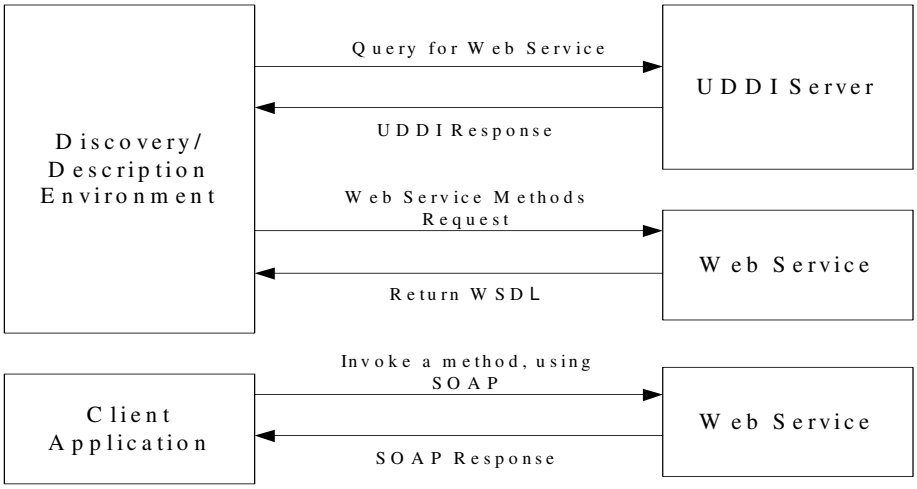

Figure 1: Web Service Technologies [2]

To call a Web service, firstly, a web service has to be discovered by querying a UDDI server. The server returns information about Web services that match the specified requirements and a link to a WSDL document that details the methods exposed by the Web service (Figure 1). The methods and parameters described by the WSDL document are used to build a SOAP request to invoke the web service.

2. PREVIOUS WORK

2.1 Concepts

Carman, Serafini, Traverso [4] propose that the complex problem of Web service composition can be simplified by breaking services into their constitute parts. An

aggregate Web service is composed of *atomic Web services*. An atomic Web service is broken into states, document types and actions that collectively identify it uniquely. Data heterogeneity is resolved by data mapping requirements that decide if data described by one data type can be substituted by another. DAML-S is used to specify schemas that describe the operations (precondition, post conditions), goals and states that are available to the system when it needs to discover and execute service operations. These schemas would semantically describe data types and enable other Web services to automatically interpret and match data types.

Sirin, Hendler, Parsia [5] present a prototype for *semi-automatic binding* of Web Services based on service constraints. They enhanced the business process flow representation of BPEL4WS with semantics to enable runtime discovery of Web Services and to resolve inter-service dependencies during dynamic binding of Web process flows. Business constraints were captured in the DAM-S ontology technology, a Semantic UDDI module was to discover the services and a dynamic binder and invoker was used. However, the Web services and their inter-dependencies still had to be known at design time.

For *composition,* Korhonen, Pajunen, Puustjärvi [6] propose that a Web service compositor should select the most appropriate available service based on policies and regulations that match the caller's organizational requirements. They describe Web services via syntactic, semantic and pragmatic (contextual), properties. They also introduce the concept of "conversation" to identify all the business documents that compose a business model. While the solution allows designers to choose Web service providers that meet organizational goals, it may be too complicated to achieve business goals.

The WebTransact framework by Tosic, Pagurek, Esfandiari and Patel [7] is a centralized approach to the *management* of distributed Web services. It enables structurally and semantically different services to work together to achieve the same business goal via composition and by integrating semantically equivalent remote services. It coordinates the sequence of service invocation within a Web service composition and manages the data flow between the services.

Automatic orchestration of services has been addressed in static environments only [11].

2.2 Related Technologies

Collaboration protocol profiles (CPPs) have been defined for ebXML, which describe the transport, security, communication protocols and business processes an organization recognizes to set up an ebXML relationship. These protocols and processes specify syntax and enable communication, but interoperability for message content, including invocations, has to be provided by additional methods. Our solution goes further by providing information for service invocation, and defining a pluggable architecture that does not need additional, connecting elements.

A number of ontological modeling and description methods are available for Web services, such as WSMO, SWSO, OWL-S, which address the same space. Some of them also have associated reasoners. Our implementation uses OWL-S markup language constructs to describe Web service capabilities and properties.

3. PROPOSED SOLUTION

3.1 Outline and Contribution

We describe a framework for automatic Web service orchestration: for dataflow management and to manage the execution of atomic units. The Web services share a stateful context to support distributed transactions, and a compensation process is available to respond to failure conditions, in particular in case of long-running transactions. The description of the services allows them to remain mobile and independent of the deployment environment. Our main contribution is the last part, dynamic orchestration that addresses issues related to the composition of services external to the choreographer, and an ability to respond to failure conditions.

3.2 Web Service Description

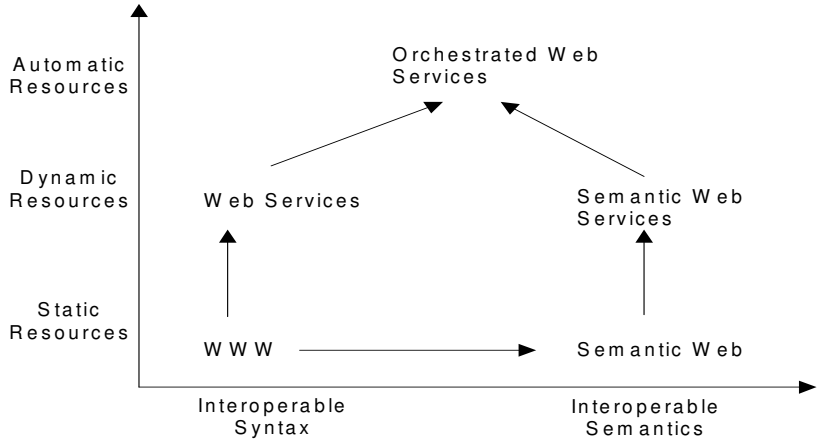

Figure 2: Convergence Models Describing Web Services

Approach

Web service description is used to compose Web services into units of work that accomplish definite goals, e.g. fulfilling a single order of a series of parts from a buyer to different suppliers. A Web service is unambiguously described by its type, and is defined by the messages exchanged between its service partners. Web service type description includes functionality as well as business semantics.

Figure 2 adapted from [8] shows that by converging the syntactic and semantics models the WWW moves from a static resource environment to a machine-processable, automatic resource environment, where functional and semantic information is described, and relationships are published and available for discovery. Figure 3 (adapted from [10]) illustrates the associated information that needs to be published to enable a collaborative Web Service Environment.

Active Information Management

In order to be able to move freely between domains, data needs to be classified and rules applied so each domain can understand not only the data, but also their relationship. XML only provides syntactic interoperability, and we added semantic

information in the form of "smarts" to the data, rather than include it in the application's processing of the data [9].

To achieve Active Management of Data we use four properties [9]: XML documents for a single domain, XML documents composed from multiple domains, XML documents that can describe relationships, and Trust Relationships.

After applying all the above, the data is application-independent, composable and classifiable. The data is tagged with information that is machine-understandable and can be extended to form an ontology.

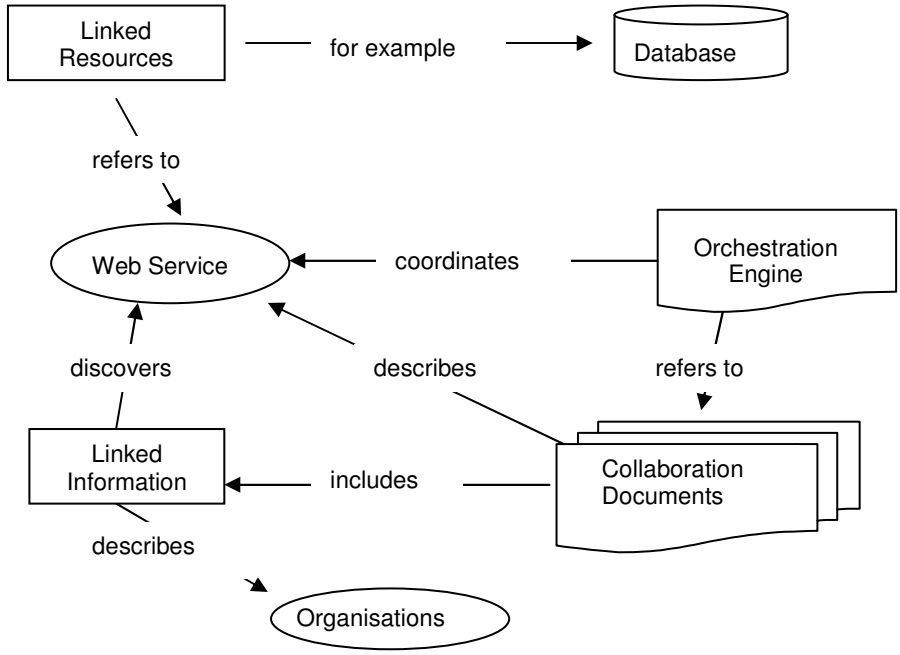

Figure 3: Advanced Web Service Model

Business Profiles

Profiles help organize and apply rules and properties of Web services within diverse application environments. To support discovery of services, Web services are abstracted to a set of high-level specifications that allow for description of profiles and a means of associating them to Web services. The profile lists the technical information needed to do business electronically, i.e. which XML schemas to use, security rules etc, and describe the relationship and dependencies between services. The profiles exist as independent documents to enable them to be composed in unique combinations and need to be accessible online.

Web Service Orchestration

To compose Web services into a structured workflow, an implementation neutral standard vocabulary (structured protocol) script is used, together with an orchestration engine to implement the process descriptions. This script is compiled into runtime scripts that are executed by the orchestration engine. The script

provides syntax for describing business workflow logic, sequences and logic, and description of public and private business protocols.

Web Service Description
We describe a Web service as an ontology, so Web service profiles become syntactically interoperable, and terms can be mapped between the participating Web services. This adds a representation and inference layer on top of the Web's current layers and enables Web services to play the different roles, such as information brokers, search agents, information filters and intelligent information integrators.

Because of using an ontology language, Web services possess knowledge representation and can behave like plug-in sockets when connecting the user to *virtual business environments*. We can also use ontological reasoners both at edit-time (model construction) and at run-time (system execution), and Web service compositors can organise individual Web services into aggregated Web services by matching business requirements with suitable Web service partners.

3.3 Capturing Web service profiles

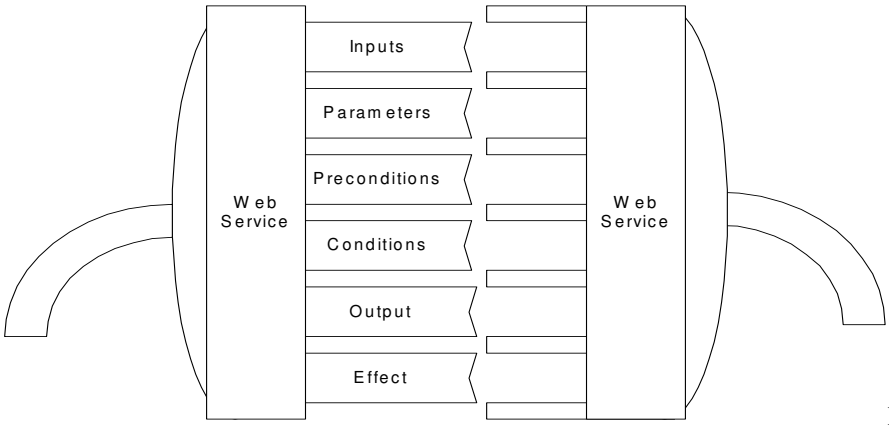

Figure 4: Web services externalising their properties to create plug-in services

To externalise a Web service, it is viewed as a black box with properties, such as preconditions, input, output etc, as shown in figure 4. The rich description of Web services enables them to connect and form virtual business environments (figure 5).

Web service workflow
The composite service workflow is described by first defining each atomic service, then by describing the composite services that aggregate the cooperating atomic services into a structured workflow. An example is ItemBuy, a composite Web service that uses the following atomic services: LocateItem, PutItemInCart, SignIn, CreateAcct, CreateProfile and LoadProfile. The Web service parameters and service workflows are stored in a repository.

3.4 Implementation

Our prototype extends the Description Layer (Web Service Technology Stack) by

adding an upper ontology layer to describe the Web services and service properties unambiguously. The advantages of this approach are:

- It complies with the Web Service Technology Stack, which is universally available.
- Existing Web service technologies such as WSDL and UDDI can be used to combine the syntactical discovery of Web services (via an UDDI registry) with the semantic description of a Web service profile Ontology.

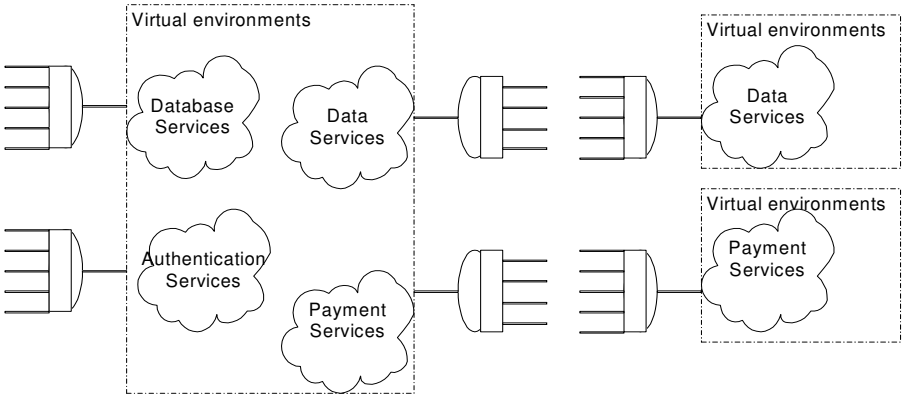

Figure 5: Web services creating business virtual environments

The profile matching is transparent to the requester. A Service Broker provided the functionality of discovering suitable Web services, by querying syntactic Web service information via the UDDI registry and semantic information via the semantic repository. The results of the queries are combined and presented to a Reasoner that mediates between various Web service profile ontologies by applying rule reasoning. This approach also provides limited fault-tolerance, and service failures result in the initiation of a new discovery process.

The combination of semantic and syntactical querying enabled service providers to publish their services via the UDDI server. The UDDI service was extended with an OWL classification identifier that also contained links to associated OWL-S documents. The Service Broker queries the UDDI server to obtain the OWL classifier, downloads the associated OWL-S document links and sends the links to the Reasoner. The Reasoner uses the links to query the services profile, identify the service types and discover if the service types are compatible. This reasoning is automatic. The rules defined were simple “is a” or “has a” rules which can be extended. The service providers must plan ahead when they publish their services by adding this special classification.

4. CONCLUSION

Web service compositions often involve external services that are outside the control of the orchestrator. This raises the problem of matching services syntactically and semantically, and responding to failures.

In our solution, automatic service discovery and composition was achieved via modelling the services property types and their messages as an ontology, which enabled a reasoner to mediate between services. To discover matching services, machines automatically query and reason the data, arbitrate between descriptions and then choose the most appropriate service, based on a given set of rules.

After detecting service failures or unavailability of services, the framework can initiate a new service discovery to readjust execution, and provide limited resilience.

A prototype model was built that combines a UDDI registry service with a semantic framework. By enhancing the WSDL descriptions, a Web service is unambiguously identified via its type. Using ontologies enabled reasoning by querying the data to identify type and its relationships. The prototype demonstrated that services could understand not only their data types but also other services' data types that they had no prior knowledge of or relationship with.

5. REFERENCES

1. Paulo F. Pires, Mário R. F. Benevides, and Marta Mattoso "*Building Reliable Web Services Compositions*". Computer Science Department. COPPE - Federal University of Rio de Janeiro, Brazil, 2002
2. James Snell, Doug Tidwell and Pavel Kulchenko "*Programming Web Services with SOAP*". O'Reilly, ISBN: 0-596-00095-2, January 2002
3. Vladimir Tosic, Bernard Pagurek, Babak Esfandiari, Kruti Patel. "*On Various Approaches to Dynamic Adaptation of Distributed Component Compositions*". OCIECE-02-02, June 2002.
4. Mark Carman and Luciano Serafini and Paolo Traverso, "*Web Service Composition as Planning*", carman,serafini,traverso@irst.itc.it, http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf
5. Evren Sirin, James Hendler, and Bijan Parsia, "*Semi-automatic Composition of Web Services using Semantic Descriptions*", University of Maryland, Computer Science Department, College Park MD 20742, USA evren@cs.umd.edu, University of Maryland, MIND Lab, 8400 Baltimore Ave, College Park MD 20740, USA hendler@cs.umd.edu, bparsia@isr.umd.edu
6. Jarmo Korhonen,Lasse Pajunen and Juha Puustjärvi, "*Automatic Composition of Web Service Workflows Using a Semantic Agent*". Software business and Engineering Institute,Helsinki University of Technology, {jarmo.korhonen,lasse.pajunen,juha.puustjarvi}@hut.fi, Proceedings of the IEEE/WIC International Conference on Web
7. Vladimir Tosic, Bernard Pagurek, Babak Esfandiari, Kruti Patel, "*On the Management of Compositions of Web Services*". Network Management and Artificial Intelligence Lab, Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada {vladimir, bernie, babak, kpatel}@sce.carleton.ca
8. The Semantic Web, A guide to the Future of XML, and Knowledge Management, Micheal C. Doaconta, Leo J Obrst, Kevin T. Smith, Wiley Publishing. Inc, 2003
9. Michael C. Daconta, Leo J. Obrst,Kevin T. Smith, Wiley Publishing, Inc. "*The Semantic Web-A Guide to the Future of XML, Web Services, and Knowledge Management*" ISBN: 0471432571.
10. Tim Berners-Lee, Original Web proposal to CERN, http://www.w3.org/History/1989/proposal.html
11. Dyaz, G.; Cambronero, M.E.; Pardo, J.J.; Valero, V.; Cuartero, F. "*Automatic generation of Correct Web Services Choreographies and Orchestrations with Model Checking Techniques"*, Telecommunications, 2006. AICT-ICIW apos;06. International Conference on Internet and Web Applications and Services/Advanced International Conference on Volume , Issue , 19-25 Feb. 2006 Page(s): 186 - 186