RC23826 (W0512-019) December 2, 2005
Computer Science

IBM Research Report

Proceedings of the IBM PhD Student Symposium at ICSOC 2005

Andreas Hanemann, Ed.
Leibniz Supercomputing Center, Germany

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Preface

In the last few years a paradigm shift in distributed computing could be witnessed. Service Oriented Computing has paved the way to changes in the design, architecture, delivery, and use of software applications. Services are defined as autonomous platform independent software elements which can be used to collaborate as distributed applications especially across organizational boundaries. They can be described, published, discovered, orchestrated, and programmed using standard protocols.

A promising way to realize the idea of Service Oriented Computing are Web Services. These services are designed for the implementation of business processes within a distributed computing environment and are based on a set of evolving standards.

In order to allow for PhD students to get involved into the current developments in the area of Service Oriented Computing and Web Services, the IBM PhD Student Symposium provides a forum for the presentation of PhD theses. PhD students get the opportunity to present and discuss the current status of their work in order to receive valuable feedback.

The focus of the IBM PhD Student Symposium is aligned to the co-located ICSOC conference (International Conference on Service Oriented Computing). Therefore, contributions with respect to the following topics were requested in the call for papers.

- Service Information Modeling
  - Service Description
  - Service Dependency Modeling
  - SLA Modeling
- Service Configuration
  - Service Composition
  - Service Discovery
  - Service Deployment
- Service Monitoring and Management
  - Service Performance
  - Service Scheduling
  - Service Fault Management
  - SLA Management
- Services Framework
  - Service Lifecycle Management
  - Service Oriented Architectures
  - Quality of Service
  - Services and Workflow/Business Processes
  - Service Semantics
- Applying Service-Oriented Architectures
  - Theoretical Application of Services

- Services in Grid Environments
- Services for Peer-to-Peer Applications
- Security and Privacy Issues with respect to Services

The review of each submission has been performed by two renowned experts in the area with respect to innovativeness and contribution, relevance to call for papers, technical content, paper organization and presentation, reference to related work, and overall recommendation.

November 2005

Andreas Hanemann
Program Chair
IBM PhD Student Symposium

Organization

The IBM PhD Student Symposium is held in conjunction with the 3rd International Conference on Service Oriented Computing (ICSOC 2005). It is organized as a whole day event on December 12, 2005 in the Mercure Hotel Amsterdam aan de Amstel (Amsterdam, The Netherlands). The homepage of the symposium can be found on the conference homepage (www.icsoc.org ⇒ PhD Symposium).

Program Committee

Program Chair:	Andreas Hanemann (Leibniz Supercomputing Center, Germany)
Committee Members:	Marco Aiello (University of Trento, Italy) Claudio Bartolini (HP Labs, USA) Schahram Dustdar (Technical University of Vienna, Austria) Rik Eshuis (Technical University of Eindhoven, The Netherlands) Andreas Hanemann (Leibniz Supercomputing Center, Germany) Alexander Keller (IBM Research, USA) Rania Khalaf (IBM Research, USA) Frank Leymann (University of Stuttgart, Germany) Massimo Mecella (University of Rome, Italy) Mike Papazoglou (Tilburg University, The Netherlands) Pierluigi Plebani (Politecnico di Milano, Italy) Thomas Risse (Fraunhofer IPSI, Germany)

Sponsoring Institutions

IBM Research, USA

Table of Contents

Session on Service Oriented Architectures

Towards an Ontology-enabled Service-Oriented Architecture 1
Maksym Korotkiy (Vrije Universiteit Amsterdam, The Netherlands)

Engineering Authorization Services for the Service Oriented Architecture 7
Sarath Indrakanti (Macquarie University, Australia)

Web Services Software Architecture 13
Syahrul Fahmy (University of Manchester, United Kingdom)

Session on Management and Security

Using an Aspect Oriented Layer in SOA for Enterprise Application Integration 19
Chinthaka D. Induruwana (University of Manchester, United Kingdom)

Towards a Service Management Information Base 25
Martin Sailer (University of Munich, Germany)

Secure Document Circulation Using Web Services Technologies 31
Shane Bracher (Bond University, Australia)

Session on Processes and Workflows

Model-driven Adapter Development for Web Services Interactions 37
Hamid Motahari (University of New South Wales, Australia)

Towards a Transaction Framework for Contract-Driven, Service-Oriented Business Processes 43
Ting Wang (Eindhoven University of Technology, The Netherlands)

A Business Aware Transaction Framework for Service Oriented Environments 49
Benedikt Kratz (Tilburg University, The Netherlands)

Modeling and Analysing Web Services Protocols 55
Julien Ponge (ISIMA, France)

Session on Quality of Service

Web Service Discovery with Implicit QoS Filtering 61
Natallia Kokash (University of Trento, Italy)

Web Service Composition Quality Modelling 67
Ganna Frankova (University of Trento, Italy)

Extending OWL for QoS-based Web Service Description and Discovery .. 73
Kyriakos Kritikos (University of Crete, Greece)

Towards an Ontology-enabled Service-Oriented Architecture

Maksym Korotkiy

Vrije Universiteit Amsterdam, Department of Computer Science
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
maksym@cs.vu.nl

Abstract. In the field of Semantic Web Services (SWS), the OWL-S and WSMF approaches provide us with ontology-based frameworks for WSDL web services. These description frameworks enable automation of high-level tasks such as discovery, invocation and composition of web services. In our work we focus on the *software architectural aspects of ontology-enabled services* to define an Ontology-enabled Service-Oriented Architectural style (Onto-SOA). The proposed style provides for a general integration mechanism for ontologies and SOA in a manner consistent with the SOA constraints. Onto-SOA is independent from an ontology language and a particular web service technology. Therefore, it can be easily combined with other styles to devise software architectures for existing SWS frameworks as well as novel approaches to integration of ontologies and SOA.

1 Introduction

The Service-Oriented Architectural style (SOA) is gaining momentum. Gartner predicts that SOA will become a prevailing software engineering practice in coming years[1]. SOA attracts industry attention by providing for a flexible and cost effective way to re-use functionality captured in loosely coupled, business-aligned services. WSDL [1] web services is the most well-known, however, *not the only possible*, implementation of SOA.

The Semantic Web [2] is a vision of the next generation of the Web that makes resources published on the Web "understandable" by machines, thus improving usability of these resources. One of the main means in achieving the vision is ontologies, the responsibility of which is to be a carrier of semantics shared among agents.

The significant potential of the combination of web services and the Semantic Web has been recognized by the Semantic Web Services research area, with OWL-S [3] and WSMF [4] approaches being the two most well-known representatives. Both OWL-S and WSMF are aimed at providing an extensive ontology-based description framework for WSDL web services. In our work we analyze the combination of ontologies and web services from the viewpoint of *software architectures* [5] and one of our goals is *to define an Ontology-enabled Service-Oriented Architectural style (Onto-SOA) to provide for a general, technology- and ontology (language) independent framework for integration of ontologies and service-oriented architectures*.

[1] http://www.gartner.com/resources/114300/114358/114358.pdf

On the more practical side, our work is motivated by the problem of applying ontologies in design of applications for the e-Science domain. In the e-Science domain a large variety of knowledge intensive computational resources exists: experiment design and execution environments, model simulation and statistical analysis services. Ontologies have a great potential to improve (re-)usability of these resources. In order to realize this potential, end-user applications have to be developed. However, at present, an application developer faces both complexity of the state-of-the-art approaches and very limited guidance available on how these approaches can be applied to development of ontology-enabled software. We address this problem by *further operationalizing the proposed Onto-SOA style into a pragmatic framework ready to be applied in ontology-enabled applications*.

The rest of the paper is devoted to a brief introduction to SOA (Sec. 2) followed by the analysis of relations between SOA, web services (Sec. 3) and Semantic Web Services (Sec. 3). During this analysis we outline a gap between SOA and current approaches to ontology-enabled web services. We address the gap by defining Onto-SOA (Sec. 4). Finally, we conclude with a summary and directions for future work.

2 Service-Oriented Architectural Style (SOA)

An *architectural style* is "a coordinated set of constraints on architectural elements and relationships among those elements within any architecture that conforms to that style" [6]. The general architectural elements are *processing components*, *connectors* and *data*. Processing components can transform data elements. Connectors provide for an abstract mechanism that mediates communication, coordination or cooperation between components [7]. From the processing component perspective, connectors transfer data without modifying it. However, internally a connector can contain a complex subsystem that can subject the data to a number of intermediate transformations.

SOA is a style that is constrained to induce a number of desirable characteristics on a compliant architecture. In the literature, SOA is often characterized as a style that *supports loosely coupled, business-aligned, networked services to enable flexibility and interoperability, in a technology-independent manner*.

Services in SOA represent coarsely-grained expertise from an application (business) domain. A service makes this problem-solving expertise available to a consumer. SOA achieves loose coupling between its processing components (a service consumer and a service) by requiring that *connectors are simple, generic and application independent*, and that *descriptive messages constrained by an extensible schema are delivered through these connectors*. A connector is application independent and generic if it does not introduce *real dependency* and minimizes *artificial dependency* between a service and consumer.

Since the connectors are generic, all application-specific semantics must be expressed in descriptive messages that communicate a problem description from a service consumer to a service provider. The messages specify what is to be solved but not how it should be solved. The reason for this is that a service provider possesses problem-solving expertise that is missing in the service consumer.

In order to enable "understanding" between communicating components, messages *must have unified syntax and structure* and *must be expressed in a vocabulary shared by communicating components*. The vocabulary is defined in a schema, extensibility of which is very important due to an open range of possible application domains.

In SOA, loose coupling is achieved by enforcing application independent connector elements. All application-specific semantics is to be contained in data elements (messages). A service can be uniquely characterized by a set of messages it can properly interpret. These messages are constrained by a certain schema that can be seen as an abstract definition of a service. From a service consumer point of view, a service is known via this abstract definition only. Since a service represents problem-solving expertise in an application domain, the schema effectively plays a role of a specification of this expertise. Such view on a schema in SOA allows us to make a natural transition from a schema to an ontology and treat an ontology as an integral component of SOA.

3 Web Services

In the literature the term *web services* most often refers to WSDL/SOAP-based services. However, in our work we employ it in more general sense to refer to *a derivation of SOA that employs standard Web transport protocols (HTTP, for example) and XML language to express messages*. The main reason for this is that WSDL/SOAP is not the only possible way to apply SOA on the Web.

WSDL/SOAP Web Services presently are the most popular application of SOA on the Web. WSDL provides a description framework for web services and is aimed at service invocation primarily. SOAP [1] provides for a standard way to structure messages that can be carried over a variety of transport protocols with HTTP being most frequently used one.

In the very beginning SOAP supported the RPC (Remote Procedure Call) communication style only[2] and later on the document-based flavor of SOAP was introduced. The document-based SOAP messages are capable of carrying any XML-based content and are not restricted to the RPC communication style only.

From the SOA perspective, RPC introduces an additional *artificial dependency* between a consumer and a service and, therefore, hinders loose coupling between these components. Furthermore, the RPC style messages tend to be prescriptive rather than descriptive: with the RPC-message a consumer prescribes a service *how to solve a problem* rather than describing *what is to be solved*. In RPC web services, message semantics often becomes workflow-dependent and that hinders integration of services. With SOAP 1.2 the RPC style has become optional, thus, enabling us to avoid it in WSDL/SOAP web services. However, RPC still attracts most of the attention in the research community.

Document-based WSDL/SOAP web services are fully compatible with SOA. Normally they rely on the XML Schema language to describe a structure of an XML document. However, XML schemas do not allow us to capture semantics of message ele-

[2] http://www.xml.com/pub/a/ws/2001/04/04/soap.html

ments. WSDL extension elements allow us to use different schema languages capable of capturing semantics, however, there is little guidance available on that.

WSDL/SOAP web services introduce a significant amount of conceptual and architectural elements [8]. A significant part of it is devoted to the RPC communication style that is to be avoided in SOA. Moreover, in many cases a document-based invocation that is performed via the well-established communication interface (HTTP protocol and the request-response message exchange pattern) is sufficient to define an operational SOA. Such a SOA can be both simpler and sufficient for many practical applications.

Representational State Transfer (REST) Web Services are based on the REST architectural style [6] that is designed around concepts such as Resource and URI. The interface among agents in REST is limited to the HTTP protocol that provides for both the transport layer and actions applicable to resources. REST web services normally employ XML to express messages and XML Schema as a vocabulary definition mechanism. However, since the infrastructure underlying REST web services is simple, it becomes rather straightforward to employ an ontology-based schema definition mechanism in REST web services. Another important feature of REST is that this style has been designed to fulfill requirements of the Web [9]. REST web services, therefore, gain many architectural properties of the Web which have already proven to be successful.

WSDL/SOAP web services require a significant infrastructure and have a high adoption barrier. On the other hand, REST web services require little infrastructure in addition to what is already provided by the Web, are fully compatible with the SOA constraints, can be easily made to employ an ontology-based schema definition languages and sufficient for many applications.

Despite the fact that REST web services are not the standard web services and less widely publicized, they are known in some cases to be preferred over WSDL/SOAP web services. For example, Amazon provides interfaces for both WSDL/SOAP and REST web services, and 85% of the usage is on the REST interface[3].

Semantic Web Services The state-of-the-art approaches to Semantic Web Services [10] such as OWL-S and WSMF employ ontologies to provide semantically rich descriptions of WSDL/SOAP services. Such descriptions provide us with meta-data about a service and can be applied to automate web service-related tasks such as discovery, invocation, composition etc. The fact that a number of related but independent tasks are addressed within a single framework makes its application cumbersome if only some of these tasks are important for a given SOA.

From the SOA perspective, SWS define a relatively complex semantic connector element responsible for establishing a connection between a service and its consumer in a way that insures semantical compatibility of messages. The connector element can perform complex operations, however, the service still operates on semantics-free messages (SOAP RPC in most cases). This makes us to believe that *the state-of-the-art approaches to SWS do not address direct exchange of semantically rich messages between processing components in SOA*. For example, if there is a web service designed

[3] http://www.oreillynet.com/pub/wlg/3005

to operate over RDF [11] messages then the-state-of-the art SWS approaches are of little use here, and if the service is not a WSDL/SOAP-based one then there is even less possibilities to apply the existing SWS approaches.

4 Onto-SOA: an Ontology-enabled SOA

We propose Onto-SOA – SOA that assumes a direct exchange of semantically rich messages between processing components. Onto-SOA is general meaning that it can be combined with any SOA-compatible architecture (WSDL/SOAP or REST) and with any ontology language. The style can be gradually refined into derivative styles that will support additional service-related activities.

We derive the Onto-SOA style from SOA by introducing an additional constraint on architectural data elements (messages): *An ontology language must be used to express a schema underlying messages*. This constraint explicitly addresses the semantical interoperability of messages. In Onto-SOA we assume that both a service and its consumer are aware about an ontology underlying messages.

Implementing an Onto-SOA with an Ontology Language To check whether the newly-derived Onto-SOA is compatible with an underlying SOA, we have to validate the introduced constraint against the general SOA requirement on schema extensibility.

In most of the modern applications of SOA, XML and XML Schema provide a unified syntax and vocabulary definition mechanism to messages. However, the XML Schema language addresses structural aspects only, leaving semantics of a defined vocabulary implicit. This leaves a problem of semantical interoperability among processing components unsolved but, on the other hand, does not restrict the flexibility in defining domain-specific vocabularies.

An ontology can provide a domain vocabulary, semantics of which is precisely defined in terms of ontological primitives of the underlying language. The ontology languages such as RDFS [11] and OWL [11] have fixed semantics and do not allow a user to extend it. The user is limited to model an application domain in terms provided by an ontology language. This means that the more restrictive an ontology language is – the less flexible an Onto-SOA based on this language becomes and, therefore, the more it diverges from the general requirements to SOA.

This allows us to conclude that in order to achieve the level of flexibility required by SOA, semantics of an ontology language must be extensible. However, at present, such languages are not available and we cannot foresee whether they will appear in future. If an extensible ontology language is unavailable then the least restrictive one (such as RDFS, for example) might provide a fair approximation.

In order to operationalize the introduced Onto-SOA, we have devised *MoRe* – a derivation of Onto-SOA (and a corresponding implementation) that combines the RDF/S languages with the elements of REST web services. *MoRe* has been applied to provide a number of services for the domain of units of measure captured in the UnitDim[4] ontology. The services are employed within the Unit Converter tool [12].

[4] Rijgersberg, H., Top, J.: UnitDim: an ontology of physical units and quantities. http://www.atoapps.nl/foodinformatics. Sec. News (2004)

5 Conclusions and Future Work

We propose Onto-SOA – an Ontology-enabled Service-Oriented Architectural style that addresses a direct exchange of semantically-rich (ontology-based) messages between a service and its consumer. Onto-SOA is independent from a particular web service technology (WSDL/SOAP, for example) and, therefore, can be employed within any SOA-compatible architecture. Onto-SOA makes no assumptions about an ontology language employed as a schema definition mechanism. This enables us to combine Onto-SOA with any ontology languages.

In our future work we intend to further analyze the applicability of the modern ontology languages such as RDF/S and OWL within the proposed Onto-SOA style. Further applications the *MoRe* framework – an operational RDFS/HTTP derivation of Onto-SOA – within the e-Science domain will allow us to validate the main ideas behind Onto-SOA and to provide additional guidance to software and ontology developers in designing an ontology- and service-enabled architectures.

References

1. W3C: Web Services: Recommendations, Specifications and Documents (WSDL, SOAP, etc). (http://www.w3.org/2002/ws)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific american **279(5)** (2001) 35–43
3. W3C: OWL-S: Semantic Markup for Web Services. (http://www.w3.org)
4. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce: Research and Applications **1** (2002) 113–137
5. Perry, D.E., Wolf, A.L.: Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes **17** (1992) 40–52
6. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, UNIVERSITY OF CALIFORNIA (2005)
7. Shaw, M., Clements, P.C.: A field guide to boxology: Preliminary classification of architectural styles for software systems. In: COMPSAC '97: Proceedings of the 21st International Computer Software and Applications Conference, Washington, DC, USA, IEEE Computer Society (1997) 6–13
8. W3C: Web Services Architecture. http://www.w3.org/TR/ws-arch (2004)
9. Fielding, R.T.: Principled design of the modern web architecture. In: Proceedings of the 2000 International Conference on Software Engineering. (2000)
10. Cabral, L., Domingue, J., et al: Approaches to semantic web services: an overview and comparisons. In Bussler, C., Davies, J., Fensel, D., Studer, R., eds.: ESWS. Volume 3053 of Lecture Notes in Computer Science., Springer (2004) 225–239
11. W3C: Semantic Web Recommendations and Specifications: RDF/S, OWL, etc. (http://www.w3.org/sw)
12. Korotkiy, M., Top, J.: MoRe Semantic Web Applications. In: Proceedings of the ESWC'05 workshop on User Aspects of the Semantic Web. (2005)

Engineering Authorization Services for the Service Oriented Architecture

Sarath Indrakanti

Information and Networked Systems Security Research,
Department of Computing, Macquarie University,
Sydney, NSW, 2109, Australia
sindraka@ics.mq.edu.au

Abstract. The service-oriented architecture (SOA) can be used to build new solutions leveraging services, to cleave together existing applications or to cleave apart existing applications. The SOA provides many benefits such as cost saving to organizations by increasing the speed of implementation of any application(s) required and reducing the expenditure on integration technologies. However, security is one of the main roadblocks for enterprises to delay development and deployment of their services. Although there are standards for providing confidentiality, integrity and message authentication for services, there is not yet a standard specification for authorization services for the SOA. We address this important gap in the area of security for the SOA. In particular, we will propose an authorization policy language as well as an authorization framework for the SOA.

1 Background

The SOA is an architectural style and its aim is to achieve loose coupling among interacting distributed software systems. The SOA can be defined as a way of designing and implementing enterprise applications that deals with the intercommunication of loosely coupled, coarse grained, reusable artifacts (services). The SOA is made up of independent services interconnected via messaging. Platform independent service interfaces are defined to invoke these services. The SOA consists of service providers and service consumers. Service providers define what the service looks like and how to invoke it through an implementation independent service interface. Service consumers use this interface to invoke the service. The SOA also provides discovery mechanism to act as an intermediary. Service providers publish their service interface using the discovery mechanism for consumers to find and invoke the service.

There are three broad categories of service use. 1) To build new solutions with services. Services enable enterprises to have independent pieces within applications that can be developed and maintained independently and also to scale-out. 2) To cleave together existing applications. Services enable connectivity Business-to-Business (B2B) and Enterprise Application Integration (EAI). Services also let us formulate business processes to enable workflows across heterogeneous environments in turn letting us tap in to the IT infrastructure, and 3)

To cleave apart existing applications. Use cases of the SOA include Supply Chain Management, Customer Relationship Management (CRM), Enterprise Application Integration (EAI), Enterprise Resource Planning (ERP), and Portal Web Sites amongst others. The SOA enables cost saving by increasing the speed of implementation of any application(s) required and reducing the expenditure on integration technologies. The SOA provides enterprise agility, to quickly respond to rapidly changing market needs and business requirements by quickly building new applications and quickly updating old applications.

2 Securing the SOA

In general, security for the SOA is a broad and complex area covering a range of technologies. At present, there are several efforts underway that are striving to provide security services such as authentication between participating entities, confidentiality and integrity of communications. A variety of existing technologies can contribute to this area such as TLS/SSL and IPSec. There are also related security functionalities such as XML Signature and XML Encryption and their natural extensions to integrate these security features into technologies such as SOAP and WSDL.

The WS-Security specification describes enhancements to SOAP messaging to provide message integrity, confidentiality and authentication. There is also work on XKMS defining interfaces to key management and trust services based on SOAP and WSDL. However, while there is a large amount of work on general access control and more recently on distributed systems authorization, research in the area of authorization for Web services is still at an early stage. There is not yet a specification or a standard for Web services authorization. There are attempts by different research groups to define authorization frameworks [1] and authorization policy specification mechanisms [2] for Web services. Currently most Web service based applications, having gone through the authentication process, make authorization decisions using application specific access control functions that results in the practice of frequently re-inventing the wheel. This motivated us to have a closer look at authorization service requirements for the SOA. We will first address the area of design of an authorization policy language for the SOA. Then we will propose an authorization framework for the SOA. Finally, we will demonstrate these authorization services using a case study in the health care domain.

3 Authorization Policy Language for the SOA [3]

Languages have long been recognized in computing as ideal vehicles for dealing with expression and structuring of complex and dynamic relationships. Over the recent years, a language-based approach to specifying access control policies has (rightly) gained prominence, which is helpful for not only supporting a range of access control policies but also in separating out the policy representation from policy enforcement. One standard authorization policy language defined

for the SOA can replace dozens of application-specific languages. Administrators save time and money because they are not required to rewrite their policies in many different languages. Developers save time and money because they need not invent new policy languages and write code to support them. They can reuse existing code. If one policy language specification is standardized, good tools for writing and managing policies for that language will emerge. Policy languages in which one can specify policies using XML have an advantage over other languages such as Ponder, as Web services based applications using the language can leverage on the benefits of XML such as inter-operability over multiple platforms in a heterogeneous environment. A policy language based on XML technology with its own namespaces and schemas is necessary in a heterogeneous environment of Web services. Also if an XML based policy language is used, standard specifications such as XML Encryption and XML Signature can be leveraged to secure and sign those policies where required. Such a policy languages policies can be specified and referred to by any service based application whether it is running on a Java based platform or the .NET platform. When designing the authorization policy language, we took into consideration the support for a range of authorization policies such as the dynamic separation of duty policy, that are likely to be required for services deployed in a commercial environment.

In particular, we will discuss the following in the dissertation: (a) Survey of authorization policy languages for distributed systems, (b) Analysis of the authorization policy language used by .NET MyServices [4], (c) Extensions to the authorization policy language used by .NET MyServices, (d) Design of XML-based authorization policy language for the SOA, (e) Design of standard APIs to the policy language and the policy engine, and (f) Implementation of the policy language and the policy evaluation engine leveraging existing XACL policy language and engine.

4 Survey and Analysis of Authorization Frameworks[5]

We have carried out a comprehensive survey of existing authorization models both for traditional distributed systems as well as authorization models built for different layers of the SOA.

We will discuss the following in this section: (a) Survey and analysis of authorization frameworks built for stand-alone systems, (b) Survey and analysis of authorization frameworks built for distributed systems, and (c) Survey and analysis of authorization frameworks built for the SOA.

5 Authorization Framework Design Requirements [6]

We will then lay out the design requirements for authorization services built for different layers of the SOA. Broadly speaking, the SOA comprises of Web services and business workflows built using Web services. These workflows are called business processes. Figure 1 shows the layers comprising the SOA.

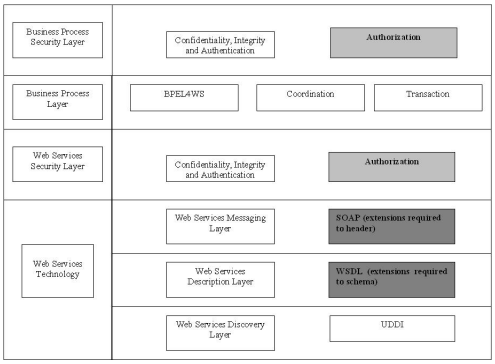

Fig. 1. Layers in the SOA

Authorization requirements differ for the Web services and business processes layers of the SOA. Authorization services for the Web services layer have special design requirements as Web services present a complex layered system. For instance, a service may be a front-end to an enterprise system and the enterprise system may access information stored in databases and files. Web services may be used by enterprises to expose the functionality of legacy applications to users in a heterogeneous environment. Or new business applications could be written to leverage benefits offered by Web services. This means an authorization architecture for Web services must support multiple models of access control. This enables legacy applications to use the access control models they have already been using as well as new Web services applications to use new models of access control or other well-known models of access control such as role-based access control (RBAC). An authorization architecture for the business process layer of the SOA must provide orchestration services to coordinate the authorization decisions from individual partners authorization policy evaluators. Each partner must be allowed to control its own authorization policies and also not require disclosing them to the entire workflow or to the workflow engine. Even in cases where the binding to actual end-points of partner services happens dynamically at runtime, the authorization architecture must be able to orchestrate the partners authorization policy evaluators and arrive at an authorization decision. Currently existing authorization frameworks are either designed for the Web services layer [1] or the business process layer [7] of the SOA. There is no unified model available that provides a comprehensive design of an authorization framework that provides authorization support to both Web services and business processes comprising the SOA.

6 Authorization Framework for the SOA

We will propose a unified authorization framework for the SOA. The framework will provide two separate architectures (indicated by the light grey colored boxes

in Figure 1) that extend the security layers of Web services (WSAA) and business processes (BPAA). The BPAA relies on the features provided by the WSAA. In particular, we will discuss the following:

6.1 Web Services Authorization Architecture (WSAA) [8]

(a) Design of the authorization administration and runtime APIs. The administration API is useful to group together a set of related Web services into collections to efficiently manage their authorization related information such as what privileges are required to be sent by a client before invoking a Web service. The runtime API is useful to invoke the necessary authorization components and receive an authorization decision, (b) Implementation of the authorization architecture APIs within the .NET framework, and (c) Benefits of the proposed architecture.

6.2 Business Processes Authorization Architecture (BPAA)

(a) Design of the authorization administration and runtime APIs. The administration API is useful to define a business process and mange its authorization related information. The runtime API is useful to invoke the necessary authorization components and receive an authorization decision, (b) Implementation of the authorization architecture APIs within the .NET framework, and (c) Benefits of the proposed architecture.

6.3 Extensions to Other Layers of the SOA

We will also describe our extensions to the Web services description and messaging layers to support the authorization frameworks for Web services and business processes (indicated by the dark grey colored boxes in Figure 1). Extensions to the description layer are required to make prospective clients aware of what authorization privileges are required to invoke a Web service or a business process. Extensions to the messaging layer are required to carry authorization related privileges, for instance, in the form of credentials.

7 Demonstration of Authorization Services in a Health Care Application

We will demonstrate the proposed authorization services the policy language and the authorization framework for the SOA using a case study in the health care domain. In particular, we will discuss the following:

(a) Demonstration of the proposed policy language and policy engine features, (b) Demonstration of the authorization services provided by WSAA and BPAA, and (c) Demonstration of the proposed extensions to the service description and messaging layers.

8 Conclusion

The service-oriented architecture (SOA) can be used to build new solutions leveraging services, to cleave together existing applications or to cleave apart existing applications. The SOA provides many benefits such as cost saving to organizations by increasing the speed of implementation of any application(s) required and reducing the expenditure on integration technologies. However, security is one of the main roadblocks for enterprises to delay development and deployment of their services. Although there are standards for providing confidentiality, integrity and message authentication for services, there is not yet a standard specification for authorization. We will address this gap in the area of security for the SOA. In the dissertation, we will first address the area of authorization policies and describe our proposal of an XML-based authorization policy language and its evaluation engine for the SOA. Then we will discuss the authorization framework requirements for the SOA. They differ for the Web services and business processes layers comprising the SOA. We highlighted the major differences in this paper. In particular, we will describe two separate authorization architectures designed using the requirements we laid out after carrying a comprehensive survey of authorization models built for stand-alone and distributed systems as well as for the SOA. Finally, we will demonstrate the authorization policy language and the authorization framework for the SOA using a case study in the health care domain.

References

1. R. Kraft. Designing a Distributed Access Control Processor for Network Services on the Web. In *ACM Workshop on XML Security*, Fairfax, VA, USA, 2002.
2. S. Godik and T. Moses. eXtensible Access Control Markup Language v1.1 (XACML), 07 August, 2003.
3. S. Indrakanti, V. Varadharajan, and M. Hitchens. Authorization Service for Web Services and its Application in a Healthcare Domain. International Journal for Web Services Research, Idea Group Publishing, vol. 2, issue 4, pages 94-119, 2005.
4. Microsoft Corporation. Microsoft .NET MyServices Specification, Microsoft Press, 2001.
5. S. Indrakanti, V. Varadharajan, and M. Hitchens. Analysis of Existing Authorization Models and Requirements for Design of Authorization Framework for the Service Oriented Architecture. In *The 2005 International Symposium of Web Services and Applications*, Las Vegas, USA, 2005.
6. S. Indrakanti, V. Varadharajan, and M. Hitchens. Principles for the Design of Authorization Framework for the Service Oriented Architecture. In *International Conference on Internet Technologies and Applications (ITA 05)*, Wrexham, North Wales, UK, 2005.
7. H. Koshutanski and F. Massacci. An Access Control System for Business Processes for Web Services. Informatica e Telecomunicazioni, University of Trento, Technical Report DIT-02-102, 2002.
8. S. Indrakanti and V. Varadharajan. An Authorization Architecture for Web Services In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Storrs, Connecticut, USA, 2005.

Web Services Software Architecture

Syahrul Fahmy

School of Informatics, The University of Manchester,
PO Box 88, Manchester M60 1QD, United Kingdom
S.Abdul-wahab@postgrad.manchester.ac.uk

Abstract. Web services have received widespread attention and acceptance in the software engineering community. Automatic composition of services, to meet user's requirements, is a powerful mechanism to ensure satisfaction of varied needs and thus enable the vision of Web services. Although Web services are equipped with the fundamental concepts and supporting technologies, the architectural style of the composite service composed from Web services is fixed by the prescriptions of the *Service Oriented Architecture*, and does not provide the level of flexibility stipulated by alternative service-based approaches such as the *Service-Based Software* vision. To enable the composition of Web services using alternative architectural styles, the author proposes that software architecture be included as a parameter in the composition process. This is enabled by the use of the Alfa framework to model the desired architectural style, and the use of this information to support service discovery.

1 Introduction

Web services are configurable software services that use open standards and protocols to connect and integrate distributed components for creating and managing computer applications. Web services have been becoming a major software trend because they provide means for integrating data sources and data transfer between applications. Web services share business logic, data and processes through a programmatic interface across a network. In order for web services to be operational, interoperability of applications over intra/internet is necessary. This interoperability requires a technology that is independent of hardware, operating system, and software specific technologies. As such, specific standards are used to address these interoperability issues that consist of XML to define structured data, SOAP as a messaging protocol, WSDL to define interfaces, and UDDI as a registry to publish available services.

There are two approaches an organization can adopt to take advantage of Web services namely the *Service Oriented Architecture* (SOA) and the *Service-Based Software* (SBS) approaches. In the SOA approach, a fixed architecture is presumed for service composition. Little flexibility, if any, is available to the service provider in determining how the service should be composed. The author argues that the service provider should be given more flexibility and 'authority' in service composition. In the SBS approach, the service provider is given this flexibility by means of a flexible

composition strategy of individual service provider, i.e. the *Know-How* capability (Fig. 1).

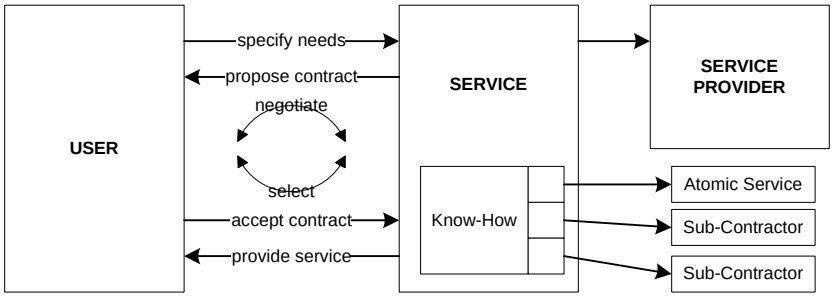

Fig. 1. Know-How Capability in Service-Based Software

This capability would enable service providers to choose the most appropriate service composition method (compose then offer services vs. offer then compose at execution). Therefore, the architecture for the software from individual services can be chosen prior to composition. To attain this capability, the software architecture style of services should be identified and determined, for example pipes and filters, prior to service composition.

In this paper the author proposes the use of the Alfa framework [1] to model the software architecture styles of services and to support the discovery of suitable services. Section 2 will present some background information on the SBS approach. Section 3 will present service architecture modeling using the Alfa framework. Section 4 will outline the expected results and contribution of this research. Finally, conclusion and future work will be presented in Section 5.

2 Background

This section will discuss some background information on Service Oriented Architecture, Service-Based Software, Software Architecture and the Alfa framework.

2.1 Software Oriented Architecture

Service Oriented Architecture (SOA) provides means to find, integrate, and use services that are available inside and outside the organization through the network/ Internet. SOA underlines three important issues for implementing Web services: interoperability, development and delivery. SOA identifies interoperability issues across open and proprietary platforms. This would allow organizations to benefit from a flexible IT architecture built on established standards without limiting the flexibility of business units to implement technologies that provide the capabilities they need. SOA also concerns the programming and development approach of services. Web

services represent a programming approach that takes advantage of available objects and content that would enable faster development cycles and lower costs due to component reuse. In addition to that, SOA also deals with the openness and flexibility of service delivery where organizations can provide access to any type of information such as inventory transactions and financial data in a reusable and open fashion.

However, SOA limits the flexibility of the service provider to choose the method for service composition. Services are composed from a predefined and rigid architecture where sometimes the user's requirements could not be fulfilled completely.

2.2 Service-Based Software

The Service-Based Software (SBS) is based on the Software as a Service (SaaS) project, carried out by the Pennine group involving academicians from three UK universities. SBS is a new paradigm proposed by the group for software engineering, a service-based approach to structuring, developing and deploying software. SBS comprises of a number of small software components that meet user requirements. When a component needs to be updated or replaced; it is disengaged; suitable replacement is searched and acquired from the marketplace; integrated; and executed. SBS are acquired and paid for on demand, as and when needed. Research efforts in SaaS have been working on the principles and practicality of SBS. An architecture for SBS has been proposed and the feasibility of the concept and primitives of the SBS architecture has been demonstrated [2]. The basics of SBS have also been demonstrated using prototypes to perform service negotiation, composition, discovery, mediation and binding [3]. Studies in SBS also include the supplier-customer relationships [4], the negotiation process [5]. A 'Negotiation Description Language' has also been proposed for SBS [6].

SBS is a better approach for Web services compared to SOA as it gives the flexibility of identifying and determining the method for services composition.

2.3 Software Architecture

Software Architecture is generally referred to as the 'blueprint' of a software system that describes its coarse-grain components. Software architecture is often defined in terms of patterns of structural organization or *Software Architecture Style* (SAS) [7]. Each style has a vocabulary of components and a set of constraints on how they can be combined. SAS provides a collection of proven solutions to recurring design problems and demonstrates a method for integrating individual style into heterogeneous structures. SAS have been used informally and explicitly, and more than one style can be used. Examples of SAS include Pipes and Filters and Implicit Invocation. In addition to the styles in [7], there are also other styles found in literature for example [8],[9],[10],[11] and [12].

Clearly, there is wealth of software architecture styles that can be used in the service composition. However, these components need to be fine-grained and formalized before it can be used in the composition process.

2.4 Alfa

Alfa is an assembly language for software architecture and supports the modeling of data, processing, and connecting elements of an architecture [13]. However, Alfa goes beyond architectural description as it is based on a small set of architectural primitives with precise dynamic semantics. Alfa also supports comprehensive style conformance analysis and architectural composition using domain-independent primitives. Alfa's characterization of styles for composing architecture is adopted from the characterization of distributed styles using five dimensions: Data, Structure, Interaction, Behavior and Topology [14]. There are seventeen styles that have been characterized in Alfa including Client-Server and Layered systems.

Alfa's primitives have been shown to be expressive enough to model any interactions involving a regular expression of input/output events on point-to-point channels [13]. As such, Alfa can be used to represent the software architecture style for the service composition process.

3 Service Architecture Modeling Using Alfa

This section will present the idea of service architecture modeling using Alfa. The architecture of Web services can be modeled using one or more software architecture styles. For example, based on a set of user's requirements, the service provider decides that the best way to fulfill the requirements is by using *Event-Based Integration* (EBI) software architecture style.

Generally, EBI refers to a system where the components can announce or broadcast one or more events. Other components in the system can register an interest in any event by associating a procedure with it. When the event is announced, the system invokes all of the procedures that have registered for the event. As such, the system would have at least 3 components namely the *Publisher, Subscriber* and *Distributor* that announces, registers for, and manages the registration of events, respectively (please refer to Fig. 2).

Using Alfa's characterization of styles and primitives, this architecture is formalized and fine-grained to computational elements in the system. These would lead to the identification and determination of the data type; structure of the system and its components; topology of the system; interactions and behavior between the components. Using topology as an example, we could determine that the *Publisher's* publish interface is connected to the *Distributor's* accept interface; the *Subscriber's* subscription interface is connected to the *Distributor's* subscription interface; and the *Distributor's* distribute interface is connected to the *Subscriber's* consume interface.

Alfa's primitives provide information on the computational elements in the components and system. For example, RELAY is a connector that contains multiple INPUT and OUTPUT portals, and enables multi-point interaction. Each portal is housed in a separate INTERFACE. Data items from each INPUT portal are forwarded to every OUTPUT portal of the RELAY. DATUMs allowed at all the portals of a RELAY are identical (words in capital are Alfa's primitives).

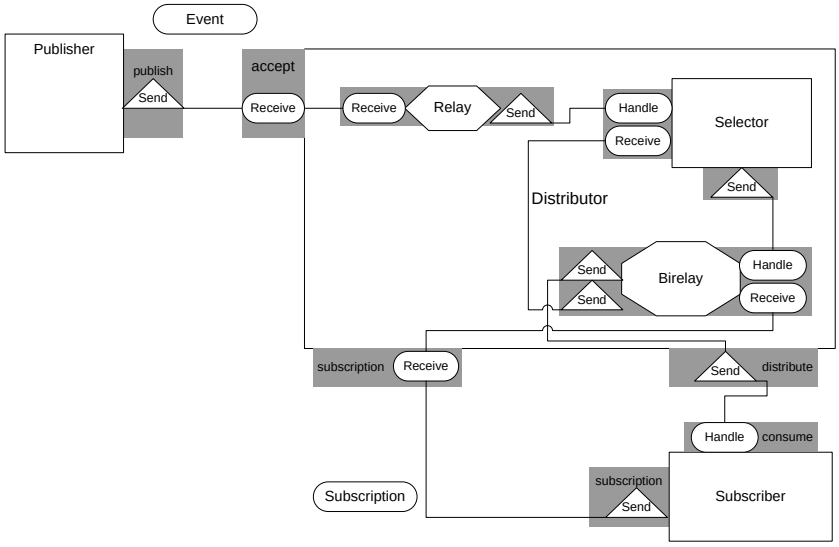

Fig. 2. Event-Based Integration Architectural Style Composed using Alfa

This information is also used for service discovery to fulfill specific requirements regarding the architecture of the service. Examples would be specific type of DATUMs, behavior of a specific PARTICLE, means of interaction either DUCT, RELAY or BIRELAY.

This section has presented an approach of using Alfa to model software architecture for the service composition process. For this approach to be implemented, software architecture and Alfa's primitives need to be included in the search criteria of Web services and the composition process.

4 Expected Results and Contribution

This research is still in an early stage where it proposes a more flexible approach to the composition of Web services by including the aspect of software architecture. In order accomplish this, at least two issues need to be addressed, a vocabulary of terms that will be used and the description of available services, both taking into account the software architecture aspects in the service composition process. Ontology of service architecture should be developed to ensure uniformity of terms used. Not only does this cover software architecture styles, but also the components in a specific style. Method for describing available services needs to be identified and developed in order for the services to be advertised and discovered. It is anticipated that the *Web Ontology Language* (OWL) and *Web Services Description Language* (WSDL) could be used for these purposes with the probability of extending WSDL to accommodate

Alfa's primitives. A prototype would be developed at a later stage to demonstrate the feasibility of the whole SBS approach for Web services composition.

5 Conclusion and Further Work

This paper has outlined the SBS approach for Web services composition. The concept and use of software architecture and Alfa in for modeling and composing Web services were presented. Work in the near future will be investigating the use of OWL and WSDL in this research as discussed in the previous section.

References

1. Nenad Medvidovic, Nikunj R. Mehta, and Marija Mikic-Rakic. (2002). A family of software architecture implementation frameworks. In Proceedings of the 3rd IFIP Working International Conference on Software Architectures , Montreal, Canada.
2. Bennett, K. H., M. Munro, et al. (2001). An Architectural Model for Service-Based Software with Ultra Rapid Evolution. Proceedings of the 2001 International Conference on Software Maintenance (ICSM 2001), Florence, Italy, IEEE Computer Society.
3. Bennett, K. H., M. Munro, et al. (2002). Prototype Implementations of an Architectural Model for Service-Based Flexible Software. Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35 2002), Big Island, HI, IEEE Computer Society.
4. Brereton, P. (2004). "The Software Customer/Supplier Relationship." Communications of the ACM 47(2): 77-81.
5. Elfatatry, A. (2002). Service-Oriented Software: A Negotiation Perspective. PhD Thesis, UMIST.
6. Elfatatry, A. & P. J. Layzell (2004). "Negotiating in Service Oriented Environments." Communications of the ACM August 47(8).
7. Shaw, M. & D. Garlan (1996). Software Architecture: Perspectives on an Emerging Discipline. New Jersey, Prentice Hall.
8. Andrews, G. R. (1991). "Paradigms for Process Interaction in Distributed Programs." ACM Computing Surveys 23(1): 49-90.
9. Berson, A. (1992). Client/ Server Architecture, Mc-Graw Hill.
10. Mettala, E. and M. H. Graham (1992). The Domain-Specific Software Architecture Program. Technical Report CMU/SEI-92-SR-9, Carnegie Mellon Software Engineering Institute.
11. Tracz, W. (1994). Collected Overview Reports from the DSSA Project. Owego, Loral Federal Systems.
12. Harel, D. (1987). "Statecharts: A Visual Formalism for Complex Systems." Science of Computer Programming 8: 231-274.
13. Nikunj R. Mehta, Nenad Medvidovic. (2003). Composing Architectural Styles from Architectural Primitives. ESEC / SIGSOFT FSE. 347-350.
14. Fielding, R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis. University Of California, Irvine.

Using an Aspect Oriented Layer in SOA for Enterprise Application Integration

Chinthaka D. Induruwana

School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road
M13 9PL
induruwc@cs.man.ac.uk

Abstract. Service Oriented Architecture (SOA) is a new method for building information systems. Approaches for integrating different application services with SOA require the client application to be hard-wired to the service. For example, if a particular service talks to another service that uses a different XML standard, then the conversion of the XML between the two services is hard-wired. In large scale integration, it is typical to find different communication standards within the integration layer. This could be the result of incremental adoption or because a company wants to protect earlier investment. Moreover, the current SOA model is not able to encapsulate the scattered concerns that crosscut services which facilitate the integration (for example, concerns including logging, security, routing, fault tolerance and transactions). Therefore, to improve maintainability and achieve better flexibility in the integration of large numbers of heterogeneous application services, we propose an aspect oriented method, in particular using an Application Service Integration Layer (ASIL).

1. Introduction

Aspect Oriented Software Development (AOSD) [1] is a new and expanding field in computer science. The software development process is driven by modelling and implementing the interactions of various software *concerns*,[1] which can be both functional and non-functional. Functional concerns can be regarded as belonging to the core application; for example, concerns that specify the behaviour of a certain application, such as the function of a class, or the method used by the class. In contrast, non-functional concerns[2] are concerns such as logging, security, routing, fault tolerance and transactions. In

[1] Ramnvias Laddad defines a software concern as: "… a specific requirement or consideration that must be addressed in order to satisfy the overall goal" [2].

[2] Non-functional concerns that crosscut are referred to as crosscutting concerns in aspect oriented terminology and *aspects* can be used to encapsulate these concerns (Figure 1b shows an example).

this paper we have called these non-functional concerns *cross-service concerns.*

Aspect Orientation (AO) is a new software engineering paradigm, the purpose of which is to encapsulate non-functional software concerns that crosscut different software components. Aspect Oriented Programming (AOP) is the encapsulating mechanism that enables AOSD engineers to program these crosscutting concerns [2, 3]. Code weaving is the mechanism that turns an AO description into code that implements the concerns.

Service Oriented Architecture (SOA) is a new method for building distributed information systems. However, there are a number of limitations with the current SOA model. Existing service software contains pre-embedded information that is typically concerned with the environment in which it is to be deployed, such as the specification of the XML that it uses to communicate with the other services (this is referred to as *integration logic* in Figure 1). This pre-embedding of information, or hard-wiring, prevents different components from being used in other external contexts. Moreover, hard-wiring leads to code tangling and scattering as shown in Figure 1a. Logging and security are examples of crosscutting concerns, and Figure 1a shows how the code of these concerns can be both tangled[3] and scattered.[4]

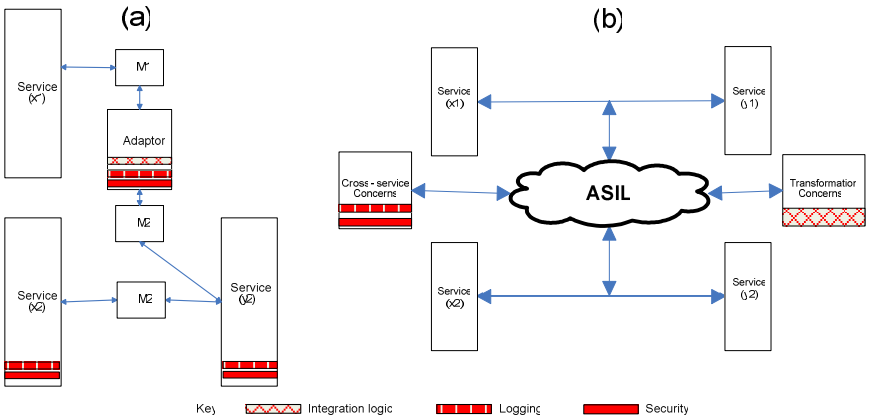

Figure 1: (a) Tangled implementation as a result of using an adaptor. (b) Application services integration layer (ASIL) overview.

In order to overcome the architectural limitations of the SOA model, there is a requirement for a new SOA model in which the services are independent of the protocols and their deployment environment. We expect

[3] Code tangling is when the code for a particular concern becomes intermixed with code for another concern.

[4] Code scattering is when the code for a particular concern becomes scattered across the system.

that an AO approach can be used to encapsulate this non-functional and crosscutting information.

This paper investigates how the principles of AO can be applied to the SOA environment. In particular, it proposes the use of an AO-layer to achieve a more flexible Enterprise Application Integration (EAI)[5] [4] solution with the benefit of encapsulated integration policies.

2. A Novel Integration Layer in Application Services

In order to facilitate the integration of cross-service concerns that support the complete decoupling of the integration logic from the application service logic, we present the *Application Service Integration Layer (ASIL)*. The motivation of ASIL is to support an AO model for EAI with the aim of having a cleaner composition model than is achievable via the current SOA model. The advantage of a layered approach is that it enables the encapsulation of cross-service concerns, as shown in Figure 1b. As a result, ASIL can be used to achieve better flexibility, reliability and scalability with respect to EAI. ASIL is designed to enable the encapsulation of both the integration logic and the cross-service concerns into separate modules. The integration logic concern deals with the different specifications and standards arising from a heterogeneous integration environment. Currently adaptors are required to enable communications between heterogeneous services (Figure 1a). This integration logic would benefit from being encapsulated separately. The advantage of encapsulating the integration logic is that, if a particular set of application services are upgraded, or another one is integrated, and the XML message specification changes, only the transformation rule at one site needs to be altered. This makes it easier to integrate heterogeneous services.

The advantage of encapsulating cross-service concerns is that it enables the non-functional concerns of the system to be handled in a uniform and consistent fashion across the enterprise. Examples of cross-service concerns are as follows:

- **Logging.** The logging concern deals with the encapsulation of the logging behaviour. When certain points of the program execution are reached, the system log is updated to store a record of the program execution.
- **Security.** The security concern deals with different security aspects of the EAI, for example, the security mechanism which is used to communicate between the different services.
- **Fault tolerance.** The fault tolerance concern deals with component failure and exception handling, among other things.

[5] Enterprise application integration can be defined as the integration of processes across third-party applications as well as legacy systems to decrease the number of adaptors which have to be developed when connecting two systems [4].

3. Case Study and High Level Design

The following case study outlines a typical set of requirements for a simple enterprise integration scenario. In Section 4 we indicate how ASIL implements this example.

1. The integration of two application services, where service x1 is using a legacy version of XML specification, and x2 is using a newer XML specification. As a result the XML messages en route between the two services need to be advised[6] by ASIL to enable communication. If this communication policy is encapsulated as an aspect, when a new service is added that also requires communication with x1, then the same advice can be applied to support this communication without the need to write another adaptor.
2. The communications between services x1 and x2 require both logging and security. The enterprise has an encapsulated aspect repository that specifies the logging and security specifications for the entire enterprise as part of the ASIL. The ASIL layer can therefore be used to advise the communications between the two services. First, the message en route has to be encrypted before it is sent by one service to the other, and then logged. ASIL has to intercept the message invocation from the receiving service, decrypt the message and log it before delivering it. The cross-service concerns of logging and security are thus applied before and after interception.

The cross-service concerns described in this example are concerns that require advising both at the service and at the message interception levels. For example, the logging concern requires advising directly at the service level. This is because, if a fault occurs within the service, preventing it from sending back a message, the log would not be updated. An example of advising at the message interception level is security, as messages en route to the services can be intercepted and then encrypted or decrypted.[7]

4. Implementation

ASIL uses an interface definition language (ASIL-IDL) which is based on AspectXML [5]. This is needed for the task of composing the different cross-service concerns in a decentralised manner. Figure 2 shows an example of an ASIL-IDL descriptor. The example shows the specification of the aspect, advice and joinpoint. The aspect is the security cross-service concern. It

[6] An advice is the behaviour of an aspect at a particular join-point (Figure 2 shows an example). A join-point is a well defined place in the structure (execution flow) of a program where the additional behaviour of an aspect can be attached (advised).

[7] ASIL supports advice at the service statically or dynamically via advising messages en route between services.

advises at the message interception level, and the advice is the WS-Security specification.

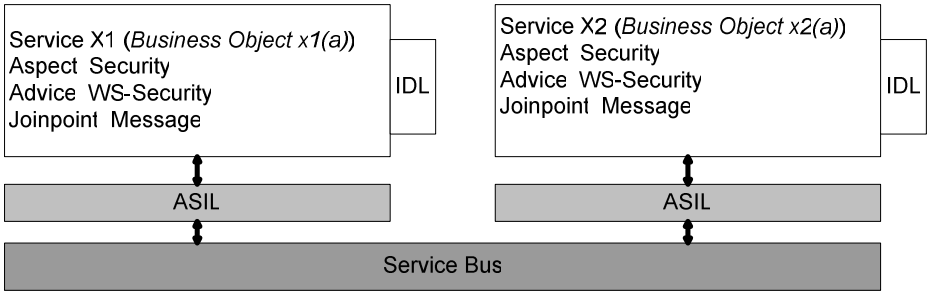

Figure 2: Interface Definition Language (IDL) Descriptor.

A detailed discussion of implementation is beyond the scope of this paper. However, a detailed description of the implementation specifications will be presented at the workshop.

5. Related Work

There are currently a number of academic and industrial research efforts trying to link AO with middleware technologies. The JBoss framework [6] is one of the most popular Application Platform Suite (APS)[8] middleware technologies. Others include J2EE, JAsCo [7], JAC [8] and Caesar [9]. However, these systems allow only implicit association of an aspect implementation.[9] Moreover, these technologies are dependent on the vendor and their aspects are restricted to the ones offered by that particular vendor. Therefore, these systems are not suitable for EAI.

There are also a number of compositional approaches;[10] these include orchestration languages, such as AO4BPEL [10]. AO4BPEL is an AO extension to BPEL4WS; it enables a more modular and dynamically adaptable web service composition. However such languages have a number of limitations. One of the main limitations is that it may not be possible for the orchestration code at a particular service to be modified. This could be due to the fact that the integration involves a legacy version of the orchestration

[8] Application platform suites are middleware technologies that are dependent on the vendor. Therefore, they only function within certain environments.

[9] By this we mean that aspects of services currently depend on a particular framework and, as a result, they have to be modified when they are reused within a different framework. With ASIL this is not the case.

[10] Service composition involves orchestration and choreography. Orchestration languages, such as BPEL4WS, describe the internal behaviour of a service within collaboration. Choreography languages, such as those used for the semantic web, can describe the interactions between services using ontologies. However, they are not mature and are not widely supported by industry.

language or because the service can not be taken offline. Another limitation of orchestration languages is that they only support static composition. A dynamic composition mechanism would be beneficial as certain services and service interactions can only be known at run-time.

The advantage of ASIL is that it is a dynamic layer which can be placed on top of heterogeneous platforms to facilitate integration. As a result it can be used to integrate legacy services, which may or may not use standards such as WSDL or BPEL4WS, whilst utilising the advantages of AO.

6. Conclusion

This paper has introduced the ASIL framework, which is an AO layered approach for EAI. ASIL allows the decoupling of concerns, such as security, from the core of the application within the SOA. The paper also suggests how a dynamic, AO, XML-based, implementation-independent pointcut language is well suited for achieving this decoupling.

7. References

1. Gregor Kiczales, J.I., John Lamping, Jean Marc Loingtie R, Cristina Videira Lopes, Chris Maeda, Anurag Mendhekar, Aspect-Oriented Programming. Proceedings of the European Conference Object-Oriented Programming, 1997: p. 220-243.
2. Laddad, R., AspectJ in Action. 2003: Manning Publications.
3. Robert E. Filman, T.E., Siobhan Clarke, Mehmet Aksit, Aspect-Oriented Software Development. 2004: Addison Wesley.
4. Sayavedra, A.L.a.L., EAI Business Drivers. EAI Journal, 2003. 2: p. 27-29.
5. AspectXML open source community project, http://www.aspectxml.org/.
6. JBoss AOP 2005, http://www.jboss.org/products/aop.
7. María Agustina Cibrán, M.D.H., Davy Suvée, Wim Vanderperren, Viviane Jonckers, JASCO for Linking Business Rules to Object-Oriented Software. System and Software Engineering Lab Vrije Universiteit Brussel, 2003.
8. A Renaud Pawlak, L.S., A Laurence Duchien, Gerard Florin, A Fabrice Legond-Aubry, A Laurent Martelli, JAC: An Aspect-based Distributed Dynamic Framework. Softw. Pract. Exper., 2004. 34(12): p. 1119--1148.
9. Mira Mezini, K.O., Conquering Aspects with Caesar. Proceedings of the International Conference on Aspect-Oriented Software Development, 2003. Boston, USA.: p. 90 - 99.
10. Anis Charfi, M.M., Aspect-Oriented Web Service Composition with AO4BPEL. The European Conference on Web Services, 2004. Erfurt, Germany.

Towards a Service Management Information Base

Martin Sailer

Munich Network Management Team
University of Munich (LMU)
Oettingenstr. 67, D-80538 Munich, Germany
martin.sailer@mnm-team.org

Abstract. The trend towards service-oriented computing has implications on the area of IT management. While traditionally management of networks and systems has dominated research as well as industrial applications, the focus is now turning to service management. One of the key concepts of integrated management is the management information base (MIB), the composite of management information. While a number of standardized MIBs have been devised in the area of network and systems management, a comprehensive MIB for service management is still missing. The PhD thesis presented in this paper aims at filling this gap by specifying semantic-rich and business-oriented service management information.

1 Introduction

With service-oriented computing becoming widely accepted by the industry, the number of commercial deployments is steadily increasing. The success of these deployments largely depends on the providers' ability to efficiently manage their IT infrastructure. To meet this requirement service providers deploy a wide array of hardware and software to deliver and manage services to their customers. Each management application deals with a certain facet of service management and therefore uses valuable data as key to its operation. The resulting situation is that information about services needs to be widely and electronically shared across departments throughout the provider's enterprise - and with partners and suppliers.

Such information is part of a Management Information Base (MIB), one of the key concepts of integrated management. The MIB contains the definition of the entities that are managed (managed objects), including their properties, their permissible behavior, and the operations that may be performed upon them [1]. Properties of managed objects are explicitly described by attributes, that usually consist of types and values. A MIB enables integration of different management applications in that it allows for exchange of semantically bound management information. For this reason, a large number of standardized managed object (MO) definitions have been devised in the area of network and systems management. However, a comprehensive MIB for service management is still missing. Hence, the realization of interoperability is impeded by the lack of a common information base for service management.

While a number of modeling frameworks for service management information have been devised, they currently show deficiencies regarding description of common service properties. The PhD thesis presented in this paper aims at addressing this gap with

the concept of a Service Management Information Base (SMIB). The SMIB approach aims at specifying semantic-rich, business-oriented service management information. This includes common service management attributes as well as association information between services and resources.

In Section 2 a typical service provisioning scenario is presented in order to derive requirements for a Service MIB. Related work in this area is discussed in Section 3 and juxtaposed to the identified requirements. Section 4 introduces the SMIB approach and presents the further steps to be taken in the PhD thesis.

2 Scenario

The Web Hosting Service offered by the Leibniz Supercomputing Center (LRZ) serves as an example IT service (see Figure 1). The LRZ operates the scientific network in Munich and hosts web sites for other research institutions. Figure 1 shows a simplified view of the service incorporating three main components: an array of web servers, a global filesystem (AFS), and a database. In addition, network components need to be considered for management (Loadb routers, Nx-switches). The management systems respectively MIBs employed by the LRZ provide means to manage networks and applications, but do not take into account services or dependencies between services and resources.Yet, that knowledge has to be deduced by systems administrators and maintained separately. It therefore often lacks formalization and is usually not reusable. Thus, a comprehensive service management repository is required to enable consistent management. The challenges stemming from that requirement can be further categorized into specification of service attributes and formalization of service resource dependencies.

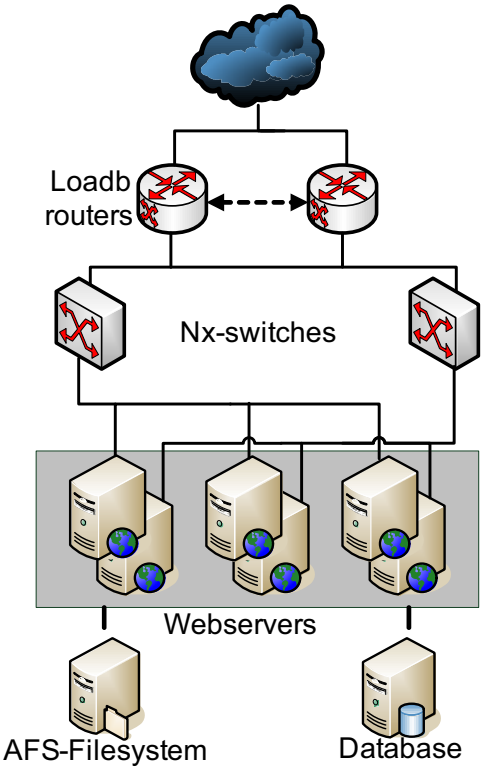

Fig. 1. Web Hosting Service at the LRZ

2.1 Specification of meaningful service attributes

Traditionally, management information has been defined from a vendors' perspective and was therefore often restricted to information that could be easily obtained by instrumenting components [2]. With the shift to service management, that perspective has to

be adapted in a way that it reflects the providers' information needs. Furthermore, services are managed along their lifecycle, in different functional management areas, thus motivating the inclusion of these aspects. Additionally, service provisioning – and consequently its management – may cross administrative boundaries. Hence, specification of service management information needs to be tailored towards the following generic requirements:

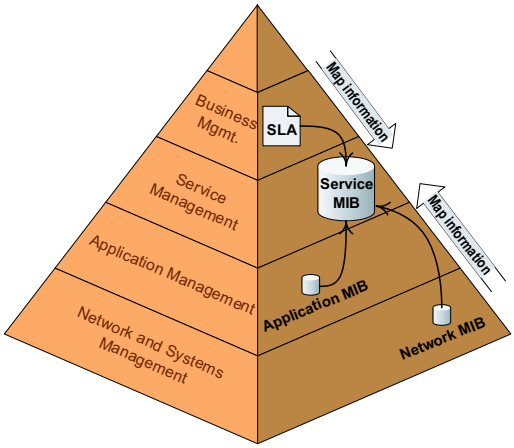

Fig. 2. Management Pyramid

Alignment to providers' information needs The MIBs currently employed contain information pertaining to applications and network elements. Examples for such data are the number of dropped IP packets, the CPU load of a webserver, or http-requests per second. While this may suffice from a network and systems management point of view, service management requires abstracted and enriched data that reflects the providers' information needs. That is, information needed to deliver the service according to the objectives set up in service level agreements (SLAs).This includes information relating to the quality level of the service, to the customers having subscribed to the service, or the impact of service failures on the business.

Administrative domains Application scenarios like dynamic service provisioning or Grid computing require services to be composed in a highly dynamic and flexible manner, thereby crossing organizational boundaries. Accordingly, the ServiceMIB must provide capabilites to encompass different management domains and therefore allow to present different views on service management information according to the needs of the involved organizations.

The functional areas of management, usually classified by the OSI FCAPS, dissect the whole complex of management. Yet, management information is shared between different functional areas, e.g., service usage information is a prerequisite for accounting as well as fault management. Consequently, all of these areas should be included in a ServiceMIB that purports to deal with service management.

The service lifecycle usually consists of a planning, negotiation, provisioning, operation/change, and withdrawal phase. Since management information is propagated and enriched among these phases, the ServiceMIB must accommodate all phases.

2.2 Formalization of dependencies between services and resources

Service management relies on knowledge regarding dependencies between services and resources. In particular, dependencies are the key to service problem resolution in that

they provide a foundation for determining causality [3]. Formalization of dependencies has to accommodate different levels (e.g., functional, organizational) as well as provide means to express complex dependencies (e.g., dependencies causing certain service quality degradations). Therefore, information contained in system, network and application MIBs needs to be combined in such a way that it reflects the service. Consequently, a formalization of dependencies must allow for translating/mapping information from other MIBs to leverage an integrated management approach (see Fig. 2).

3 Related Work

In this section prominent representative work in the area of management information modeling is presented and its significance in the focus of this paper is discussed.

Common Information Model (CIM) The Common Information Model [4] introduced by the Distributed Management Task Force is an object-oriented information model that aims at providing a common way to represent information about networks and systems as well as services. The CIM Core Model contains the specification of the class *service*, which acts as a superclass from which specific service classes (e.g. print service) can be derived. Additionally, dependencies between services and other CIM elements (e.g., *ServiceAffectsElement*) are expressed by association classes. However, only few service attributes have been defined so far for the basic service class as well as associations pertaining to that class. CIM constitutes an comprehensive modeling framework that could be extended to meet the requirements delineated in Section 2. Since it accomodates network, application, and service management information, it offers a foundation for modeling dependencies between those areas.

Web Services Distributed Management Two complementary specifications have been presented by the Oasis Web Services Distributed Management technical committee: Management Using Web Services (MUWS) deals with providing a common management interface for distributed IT resources, whereas Management of Web Services (MOWS) introduces the application of the former onto Web Services [5,6]. Service Management information is exposed through manageability capabilities and categorized into common and resource-specific capabilities. At present, only basic capabilites like the *CurrentOperationalState*, the *LastResponseTime* or *NumberOfRequests* have been defined. Moreover, the WSDM approach does not explicitly address dependencies between services and resources.

NGOSS SID As part of the New Generation Operations Systems and Software (NGOSS) program the TeleManagement Forum released a Shared Information/Data (SID) model [7]. It employs an object-oriented modeling approach and defines a framework for defining service management information from a business-oriented point of view. SID offers a promising modeling framework for service management information that builds upon sound design principles. However, it also shows deficits regarding the description of common service attributes.

Internet Information Model The IETF model revolves traditionally around the Simple Network Management Protocol (SNMP) [8]. This is a read/write, state based protocol for altering variables in a management information base (MIB) and is used by the vast

majority of commercial products. Despite of newly introduced RFCs that deal with the specification of service management information for WWW services[9], the Internet Information model does not offer object-oriented modeling features, and is therefore not suitable for modeling of service management information.

4 Approach

The Service MIB approach aims at specifying semantic-rich, business-oriented service attributes as well as association information between service and resources. It therefore consists of two main steps, that are aligned with the challenges presented in Section 2:

Specification of business-oriented service attributes In recent time, service management has been influenced by the strong trend towards process-orientation. Process-oriented frameworks contribute to that trend in that they provide guidance for enterprises striving to deploy IT service management processes. A prominent example for such a framework is the IT Infrastructure Library (ITIL). ITIL has been introduced by the British Office of Government Commerce (OGC) and provides a collection of best practices for IT processes in the area of IT service management [10,11]. Service management is described by 11 modules which are grouped into Service Support Set and Service Delivery Set. These modules describe processes along with its goals and activities as well as necessary databases and interfaces.

To facilitate the definition of service attributes we leverage a use-case driven approach. The main idea is to analyze ITIL service management processes in order to derive common service attributes. Since such processes essentially represent a specification for service management tasks, they provide a comprehensive collection of use cases that can be further utilized to derive service management information needs. This approach offers considerable benefits with respect to the requirements denoted in Section 1. Since ITIL constitutes a business-driven approach to IT service management, it already takes lifecyle and interorganizational aspects into account, while at the same time offering a sufficient level of genericity.

Example: The Incident Management Process deals with restoring normal service operation. One of the core activities within this process is the classification of incidents, consisting of e.g., priorization of incidents. To support this activity, service management information relating to the business impact of the incident is required. That is, the SLA associated with the service, the impact on related services, or affected users.

Formalization of service-resource dependencies Formalization of association information regarding services and resources requires high expressive power that cannot be expressed by single attributes. Instead, a more complex formalism needs to be employed that allows to specify associations on a per-attribute basis in a functional manner. This includes means to map information defined for network and systems to ServiceMIB attributes. Therefore, this issue has been approached with a declarative language to specify service-resource dependencies, called Service Information Specification Language (SISL). This concept has been presented in [12], which introduced a monitoring architecture that facilitates the creation of service management information from raw monitoring data according to a SISL specification.

Further Work

Previous work in the context of the incident and problem management processes has analyzed use-cases in the area of service-oriented event correlation and impact analysis [3]. This work has shown applicability of the approach and will be extended towards other ITIL processes. While up to now, basic examples of SISL for expressing the operational state of services have been drafted [12], further work on that topic will target modeling of more complex scenarios.

Acknowledgment

The author wishes to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web–server is located at http://www.mnm-team.org.

References

1. Black, U.: Network Management Standards - The OSI, SNMP and CMOL Protocols. McGraw-Hill (1992)
2. Hegering, H.G., Abeck, S., Neumair, B.: Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application. Morgan Kaufmann Publishers, ISBN 1-55860-571-1 (1999) 651 p.
3. Hanemann, A., Sailer, M., Schmitz, D.: Assured Service Quality by Improved Fault Management — Service–Oriented Event Correlation. In: Proceedings of the 2nd International Conference on Service–Oriented Computing (ICSOC04), New York City, NY, USA, ACM SIGSOFT and SIGWEB, ACM Press (2004) 183–192
4. Distributed Management Task Force (DMTF): Common Information Model (CIM) Version 2.9. Specification (2005)
5. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1 (2004)
6. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Distributed Management: Management of Web Services (WSDM-MOWS 1.0) (2004)
7. TeleManagementForum: Shared Information/Data (SID) Model Concepts, Principles, and Domains. Technical Report GB 922 Member Evaluation Version 3.1 (2003)
8. Case, J., McCloghrie, K., Rose, M., Waldbusser, S.: RFC 1902: Structure of management information for version 2 of the simple network management protocol (snmpv2). RFC, IETF (1996)
9. Hazewinkel, H., Kalbfleisch, C., Schoenwaelder, J.: RFC 2594: Definitions of managed objects for www services. RFC, IETF (1999)
10. Office of Government Commerce (OGC), ed.: Service Support. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK (2000)
11. Office of Government Commerce (OGC), ed.: Service Delivery. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK (2001)
12. Danciu, V., Sailer, M.: A monitoring architecture supporting service management data composition. In: Proceedings of the 12th Annual Workshop of HP OpenView University Association. Number 972–9171–48–3, Porto, Portugal, HP (2005) 393–396

Secure Document Circulation Using Web Services Technologies

Shane Bracher*

Bond University, Gold Coast QLD 4229, Australia
Siemens AG (Corporate Technology), Otto-Hahn-Ring 6, 81739 Munich, Germany
sbracher@student.bond.edu.au

Abstract. This paper discusses two issues. The first is the need for model driven security for Service-Oriented Computing environments to address the lack of support for specifying the security requirements of Web Services during the earlier phases of the development process. The second issue revolves around supporting secure document circulation in inter-domain, decentralized environments, and specifically, how to use model driven security and Web Services technologies to realize the design and implementation of this respectively. It is aimed that addressing both of these issues will contribute toward providing security for inter-organizational workflows which spread across multiple domains.

1 Introduction

Service-Oriented Computing has experienced considerable momentum in recent years as a new approach towards distributed computing. In this paradigm, distributed interacting entities are implemented as *services*. These are simply autonomous, platform-independent software components with the purpose of providing interoperability and collaboration within heterogeneous environments. Designing and implementing such an infrastructure can be achieved using Web Services. A strong advantage of Web Services is that it provides a standards-based, loose-coupling approach for combining multiple services (offered by different organizations) into a single, more sophisticated, value-added, composed service. The formation of these composed services is realized in terms of workflows (whereby control-flow and data-flow is specified).

A serious concern of Web Services is security. Increasing this concern is the fact that workflows can spread across domains. This complicates the issue as the security context now shifts from a single, centrally administrated domain to an inter-domain, decentralized environment. Although there is much work in existence for Web Services Security - such as WS-Security and the Web Services Security Stack [8], SAML [7], and XACML [10] - this is based on the implementational level. As for the design phase, and specifically "early Service-Oriented Computing design", support for security is lacking.

* Currently completing internship at Siemens Corporate Technology.

A promising approach is model driven security. The objective here is recognition of non-functional requirements (such as security) early in the development process. Considering such requirements at a later stage makes it (1) harder to integrate security measures into the product, and (2) increases the potential for greater security vulnerabilities arising in the product. By taking a model driven approach, this has the benefit of representing security requirements in an implementational-independent fashion, as well as providing the possibility for use with model checking tools for performing verification. Therefore, extending this model driven security approach to Web Services, and Service-Oriented Computing in general, would certainly be a step closer to addressing the security concerns of Web Services.

To demonstrate how Web Services Security can leverage the benefits of model driven security, the case of secure document circulation in inter-domain, decentralized environments is put forward. What is interesting about this case is that both it and Service-Oriented Computing share the same context - that is, distributed, heterogeneous, decentralized environments. As a result, the security issues here are comparable. Given the similarities, the question is can we achieve the objective of secure document circulation in inter-domain, decentralized environments by using Web Services technologies? More precisely, the goal is to determine if the work done towards Web Services Security can be applied to the securing of inter-organizational document flow (with the added intention of applying model driven security during the design phase).

Workflow is also an important component of secure document circulation, and this too is an issue of Service-Oriented Computing (in particular, service composition). Although this issue will be investigated in the thesis, the current focus is on security. Similarly, it is anticipated that the work of Web Services can also be applied in this regard (for example, the technologies for Web Services Orchestration).

The reminder of this paper is organized as follows: Section 2 discusses related work in model driven security; Section 3 provides further details on the research problem; and finally, Section 4 discusses future work for the thesis and concludes the paper.

2 Related Work

The issue of model driven security in the context of access control infrastructures is currently being explored by Basin et al. [2]. Their objective is to use modeling techniques to specify the security requirements of access control systems and then to automatically generate such systems from the models. The modeling language proposed is SecureUML [5] - a Unified Modeling Language (UML) based language for modeling Role Based Access Control (RBAC) policies. Much work has been done on using SecureUML for static UML diagrams (specifically, class diagrams), but as for supporting dynamic diagrams (e.g. use case diagrams, activity diagrams and sequence diagrams), this appears to be absent. For applying access control restrictions to workflows (e.g. for making flow

path decisions), support for access control policies in dynamic UML diagrams is necessary in order to specify this. Furthermore, this work limits itself to only access control and not other security properties.

The SECTINO project [4] takes an alternative approach by proposing a model driven security architecture. This architecture consists of a set of "model views" which are aimed towards representing workflows and associated security properties, but on an abstract level. The SECTINO project also focuses on UML and provides support for specifying confidentiality, integrity and non repudiation properties. Furthermore, automated "model to code transformation" is a key goal of this project. Although a broader spectrum of security properties are supported, current work focuses on class diagrams and activity diagrams. The fact that support for use case diagrams is lacking is significant given that the previously mentioned diagrams are derived from use case diagrams.

A promising development towards model driven security is the work of Jrjens et al. on UMLsec [6]. This is an extension to UML which allows for security requirements to be integrated into UML models for security-critical systems. The goals of UMLsec are to consider security requirements from the early design phases and to enable UML models to be evaluated for vulnerabilities. Hence, automated verification is a key aspect of this work. To model security requirements, UMLsec uses a set of pre-defined stereotypes and tags to label components in the model. Although consideration is provided for use case specifications in this work, further support for use case diagrams would be advantageous - especially for signifying the "scope" of the requirements.

3 Problem Description

For model driven security, this research aims to investigate existing works relating to extending the UML specifications with security specific elements so that security goals and security measures can be represented within UML models. The initial desire is to look at how the UML use case diagram can be augmented with security-based model elements. This will involve studying current proposals to determine what is missing in terms of security for Service-Oriented Computing environments, and then where necessary, proposing new security extensions to address this issue.

The types of security concerns that are of particular interest for modeling include message-level security, access control restrictions and outsourced security measures. As other UML diagrams are often derived from the information contained in use case diagrams, the objective is to formulate security constructs so that they can be easily propagated to subsequent UML diagrams (such as class diagrams, activity diagrams and sequence diagrams). Hence, this allows for reusability and consistency of the security constructs in the UML model.

The current vision for describing security requirements at the level of use case diagrams includes the following:

- Specifying security at the level of systems (all use cases must follow the requirements stated at this level).

- Specifying security at the level of use cases (defines the requirements which are specific to the given use case within a subsystem).
- Specifying security issues which are handled by trusted third parties. For example, authentication of users may be outsourced to identity providers (in order to support single sign-on).
- Specifying security requirements on messages related to externally handled security issues. For example, for an authentication token to be accepted, it must be digitally signed by a recognized identity provider. Essentially, this allows us to state the security requirements of the systems interface.

It is important to emphasize that this vision considers a Service-Oriented Computing environment. Therefore, this serves towards addressing the concern of a lack of security support for the design of Web Services during the development cycle. For added support, a further step of this work could look at verification techniques for verifying that the modeled security measures meet the security goals of the service. This would effectively provide security assurances that the service is resistant to certain attacks. Experimenting with security verification tools such as the AVISPA toolset [1] and its associated specification language, HLPSL [3], is one possibility for conducting this.

To apply these ideas of model driven security for Service-Oriented Computing, the case of supporting secure document circulation in inter-domain, decentralized environments will be explored. The objective here is to investigate a new approach for providing enhanced document protection designed for distributed, heterogeneous, decentralized environments. In terms of security, this is a non-trivial issue as multiple domains are involved. Therefore, we cannot rely on traditional security mechanisms which assume the existence of a central administration authority. As an alternative approach, security requirements will be enforced by mechanisms embedded within the document itself. The vision is the realization of a "smart" or "intelligent" document - one with richer capabilities regarding security and workflow, all within an inter-domain context.

In relation to implementing access control, this has suddenly become complicated due to the absence of a single enforcement point. In fact, the only way to provide access control in this situation is through encryption. This now results in a shift in policy specifications from "what actions is a user allowed to perform" to "who has access to the decryption key". This increased reliance on encryption introduces the need for finer granularity for confidentiality and data integrity of the document's content. Furthermore, it also introduces the additional challenge of needing to provide key protection strategies.

The current proposed framework for the smart document is displayed in Figure 1. This is simply a high-level, implementation-independent view of the smart document design. At the core of this framework are the document and document metadata components. The document component represents an ordinary document containing content. The document metadata component provides the document with, amongst other things, enhanced security capabilities. Workflow and state awareness are also planned to be provided by this component. Associated with the document metadata component are four sub-components. In

order, these sub-components provide flow awareness, state awareness, security policies and additional attributes required for fulfilling the security and workflow functionalities. The description components attached to the document and document metadata components provide structure and content control.

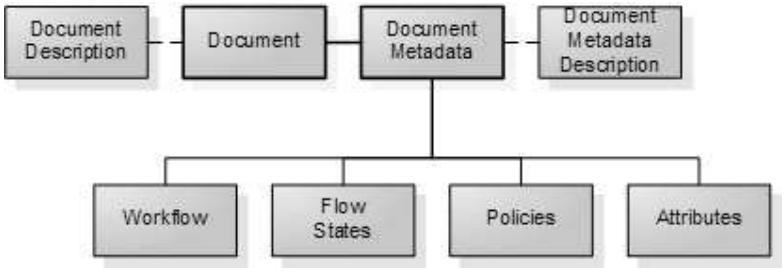

Fig. 1. High-level framework view of the smart document design.

4 Future Work and Conclusion

On the completion of the discussed work on model driven security and the design for the smart document framework, the next step is to see how Web Services technologies can be used for implementing the smart document design. For implementing the security functionalities, current technologies of interest include XML-Encryption [11], XML-Signature [12], SAML [7], XACML [10] and WS-SecurityPolicy [9]. However, this is only a preliminary list and no doubt additional technologies may be needed. Using the above mentioned technologies, the aim is to implement finer granularity document confidentiality and data integrity, decentralized access control enforcement, and security policies. After an implementation prototype has been completed, the final step will be to conduct a case study to show a real world application of secure document circulation using Web Services technologies.

In conclusion, the objectives of this research are two-fold. Firstly, it aims to address the need for security requirement specification support during the earlier stages of the development process. In particular, the desire is to focus on Service-Oriented Computing environments for this. Secondly, this research aims to apply the first objective into a more practical scenario - namely, using Web Services technologies to implement secure document circulation in inter-domain, decentralized environments. Given that the context and the issues affecting this scenario and Service-Oriented Computing are quite similar, it seems worthwhile to envision the design and implementation of this scenario in terms of Web Services technologies.

Acknowledgements

I would like to thank my supervisors Dr Paddy Krishnan, Dr Jorge Cuellar and Dr Zheng da Wu for their assistance and guidance towards my PhD candidature to date. Also, I thank my workgroup at Siemens Corporate Technology for their support and feedback on my research - particularly to our visiting colleague, Dr Shmuel Tyszberowicz of Tel Aviv University, for the many helpful discussions on model driven security.

References

1. Automated Validation of Internet Security Protocols and Applications. `http://www.avispa-project.org/`. September 6 2005.
2. D. Basin, J. Doser, and T. Lodderstedt. Model Driven Security: from UML Models to Access Control Infrastructures. *ACM Transactions on Software Engineering and Methodology*, 2005. to appear.
3. Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In *Workshop on Specification and Automated Processing of Security Requirements (SAPS 2004)*, 2004.
4. M. Hafner, R. Breu, and M. Breu. A Security Architecture for Inter-Organizational Workflows: Putting Security Standards for Web Services Together. In *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*, pages 128–135, 2005.
5. T. Lodderstedt, D. A. Basin, and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441, London, UK, 2002. Springer-Verlag.
6. G. Popp, J. Jürjens, G. Wimmel, and R. Breu. Security-Critical System Development with Extended Use Cases. In *APSEC '03: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference*, page 478, Washington, DC, USA, 2003. IEEE Computer Society.
7. OASIS Security Assertion Markup Language (SAML) Version 2.0. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`. March 15 2005.
8. Security in a Web Services World: A Proposed Architecture and Roadmap. `ftp://www6.software.ibm.com/software/developer/library/ws-secmap.pdf`. April 2002 Version 1.0.
9. Web Services Security Policy Language (WS-SecurityPolicy). `http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf`. July 2005 Version 1.1.
10. OASIS eXtensible Access Control Markup Language (XACML) Version 2.0. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`. February 1 2005.
11. XML Encryption Syntax and Processing. `http://www.w3.org/TR/xmlenc-core/`. December 2002.
12. XML Signature Syntax and Processing. `http://www.w3.org/TR/xmldsig-core/`. February 2002.

Model-driven Adapter Development for Web Services Interactions

Hamid Reza Motahari Nezhad[1,2]

[1] The School of Computer Science and Engineering,
The University Of New South Wales
Sydney, NSW 2052, Australia
hamidm@cse.unsw.edu.au
[2] National ICT of Australia (NICTA)
Sydney, NSW 1430, Australia

1 Introduction

Application integration has been one of the main drivers in the software market during the late nineties and into the new millennium. The typical approach to integration and process automation is based on the use of adapters and message brokers [1, 13, 5]. Adapters wrap the various applications (which are in general heterogeneous, e.g., have different interfaces, speak different protocols, and support different data formats) so that they can appear homogeneous and therefore integrated more easily.

Web services have emerged as a solution to (or at least as a simplification of) the integration problem [1]. The main benefit they bring is that of standardization, in terms of data format (XML), interface definition language (WSDL), transport mechanism (SOAP) and many other interoperability aspects. Standardization reduces heterogeneity and makes it therefore easier to develop business logic that integrates different (Web service-based) applications. While standardization makes interoperability easier, it does not remove the need for adapters. In fact, although the lower levels of the interaction stacks are standardized, different Web services may still support different interfaces and protocols. For example, although two driving direction services may support XML and use SOAP as transport mechanism, they may still provide operations that have different names, different parameters, and different business protocols (that is, different constraints on the order in which they should be invoked for the clients to achieve a certain functionality).

The need for adapters in Web services comes from two sources: one is the heterogeneity at the higher levels of the interoperability stack (e.g., at business protocol level), and the other is the high number and diversity of clients, each of which can support different interfaces and protocols, thereby generating the need for providing multiple faces to the same service.

In this paper, we elaborate on the problem of developing adapters for Web service interactions. In the following, first, we characterize the problem of adaptation by identifying and classifying different kinds of adaptation needs in Web services (Section 2). Then, we briefly explain our preliminary contribution on

adapter development for Web services (Section 3.1). Section 3.2 highlights the open problems, plans and research directions. Finally, Section 4 discusses related work.

2 Characterizing the Problem of Adapter Development for Web services

Interoperability among Web services, just like interoperability in any distributed system, requires that services use the same (or compatible) protocols, data formats, and semantics. Broadly speaking, we identify the following integration layers:

- **Messaging**. Any interaction requires information to be transferred among two or more parties. Hence, services should support messaging protocols interoperability, regardless of the syntax and semantics of the information. In Web services, the most common protocol at this layer is SOAP.
- **Basic Coordination**. The coordination layer is concerned with requirements and properties related to a set of message exchanges among two or more partners. For example, two or more services may need to coordinate to provide atomicity based on 2-Phase commit [1]. WS-Transaction is an example of specification at this level.
- **Business-level interfaces and protocols**. The above layers are concerned with the transfer of messages among services. To interact, services must also have compatible:
  - Interfaces (i.e., the set of operations supported by services). WSDL is the common standard for expressing interfaces.
  - Business protocols (i.e., the supported message exchange sequences by the service). These can be expressed for example using BPEL abstract processes, WSCI, or other protocol languages (see, e.g., [4, 1]).
- **Policies and non-functional aspects**. The definition of a service may include policies (e.g., privacy policies) and other non-functional aspects (e.g, QoS descriptions) that are useful for clients to understand if they can/want interact with the service.

Standardization efforts in Web services reduces the heterogeneities between service specifications, and thereby the number of required adapters. Nonetheless, standardization does not address all interoperability problems and does not remove the need for adaptation. One typical interoperability issue arises by standard evolution, so that services, which are using different versions of the same standard may not be able to interoperate. For instance, SOAP 1.2 allows the use of HTTP GET method, while SOAP 1.1 supports only HTTP POST method. This change is in the syntax of the specification, but it affects the interoperation of interacting services. Ryan et. al. [2] study this problem and employs a translator to translate the message from one format to another. The interoperation problem at the higher integration layers, e.g., business-level interfaces and protocols is more discernible. What standardization provides at these layers is a set

of languages to define service specifications and hence leaves these specifications open to syntactical, structural and semantics heterogeneities, even though they are described using the same languages. Since interoperation at the other layers has been the subject of other research (e.g., [2]) or standardization, we focus on interoperability issues and adapter development at the business-level interfaces and protocols layer.

We classify the need for adaptation in Web services in two basic categories: adaptation *for compatibility* and *for replaceability*. The first category refers to wrapping a Web service so that it can interact with another service. It is needed when two services are functionality-wise compatible, but incompatible in interface or protocol specifications. The second category refers to modifying a Web service so that it becomes compliant with (i.e., can be used to replace) another service (see [4] for detailed discussion on compatibility and replaceability).

3 Model-driven Adapter development for Web services

The intended benefit of model-driven adapter development is to help programmers to develop adapters through a methodology and semi-automated code generation, starting from the protocol definitions. In the following, we first explain our preliminary contribution in this direction and then outline the open problems and future work.

3.1 Capturing Mismatches between Protocols as Design Patterns

Essentially, an adapter is a service which sits in between incompatible services and compensate for their differences [13]. This requires performing activities such as receiving and storing messages, transforming message data, and invoking service operations. These tasks can be modeled by process-centric service composition languages such as BPEL. We have analyzed interfaces and protocols of Web services to identify the most typical differences and for these we have captured the corresponding mismatches in form of mismatch patterns, which are design patterns [3]. Each mismatch pattern includes an adapter template, which is represented as a BPEL process skeleton to handle the mismatch, as well as a sample usage. The template can be used both as guideline for adapter developers and as input to a tool that automatically generates the adapter code.

We distinguish between two types of mismatches: *interface-level* and *protocol-level*. Mismatches at the interface-level characterize heterogeneities related to operation definition in WSDL interfaces. Examples at this level include operation signature mismatch and parameter constraint mismatch. The former is concerned with identifying differences and providing mappings between the signatures of operations (e.g., operation names, name and type of in/out messages and and their parameters), and the latter deals with potential mismatches between the constraints on parameters in the input/outputs messages (e.g., accepted value ranges). Protocol level mismatches characterize heterogeneities related to message choreography and temporal/transaction properties. Examples

include differences on the order in which each protocol accepts a certain message, there is an extra (missing) messages in one protocol, or the case where the information required in one message in one protocol is captured in several messages in the other and so there is a need for a merge/split action.

Given the mismatch patterns, an analyst can identify the differences between business protocols by instantiating the mismatch patterns as many times as required. Currently, we are developing mechanisms to compose adapter templates. We are also implementing all research on adapter development using Java/J2EE as plug-ins in the Eclipse platform. These include a mismatch pattern editor and an adapter code generator, which generates BPEL code.

3.2 Towards an Extensible Framework for Model-driven Adapter Development

We believe the effective use and widespread adoption of service technologies and standards requires high-level frameworks and methodologies for supporting automated development and interoperability. The requirement of such a framework are as the followings:

- **(Semi-)Automatic identification of mismatches**. Our research so far does not address the problem of (semi-) automatically identifying differences between two business protocols. This problem is extremely challenging as any solution to it must take into account advances in multiple research disciplines including software component matching and adaptation [14, 13], protocol algebra for Web services [4] and schema matching [12] and semantic Web services [15–17]. This problem could be studied at two levels:
  - **Interface level**. The goal is to (i) (semi-)automatically identify mismatches between WSDL interfaces of services in terms of identified mismatch patterns, (ii) generate relevant mappings between operations and messages in the interfaces to be the input for the adapter templates in the mismatch pattern. It is a very challenging problem, and a variation of it has been the subject of other research [6, 9, 7], although they do not consider service interfaces with the constraint of protocol models.
  - **Protocol level**. The goal is to be able to semi-automatically discover mismatches between two protocols. We use an extended state machines formalism to specify protocols [4]. Protocols also can be represented using citeabstract process notion of process-centric languages such as BPEL. The research at this level is mostly related to adapter development for protocols in software components [13] and model-driven service development [10]. Our preliminary investigation into semi-automatic identification of mismatch shows that the problem of finding mismatches at the protocol level would be considerably simplified by identifying the mappings at the interface level. So, an approach to tackle this problem is to first elaborate on the problem of generating mappings at the interface level and then extend it to the protocol level.

– **A methodology to guide semi-automated adapter development**. There is a need for an extensible model-driven framework and a methodology to generate adapter code from given protocol models. The framework should be extensible to allow for interpolating of variant methods for interface and protocol mismatch identification, when they become available.

4 The State of the Art in Adapter Development

There has been substantial efforts and progress in in the area of Web services most of which has been focused on service description models and languages, standards, and on automated service discovery and composition [1]. Recently, researchers have considered the problem of similarity and compatibility at different levels of abstractions of a service specification (e.g., [4, 8, 6, 9, 7, 11, 16]).

In [6] and [9], techniques for assessing the similarity of WSDL interfaces of services are proposed for service discovery but these do not consider the problem of identifying interface mappings. In [7], a framework for handling differences among service interfaces is proposed, however, the assumption that Web service interfaces are derived from a common base limits its application to real world scenarios. This problem of identifying interface mapping is akin to that of identifying correspondence and mappings in schema matching [12] with the difference that in schemas there is no notion of messages and also the operation abstraction to relate them.

In terms of protocols specification and analysis, existing approaches provide models (e.g., based on pi-calculus or state machines) and mechanisms to compare specifications (e.g., protocols compatibility and replaceability checking) [4, 8]. Considering from the software engineering perspective, in [14], the authors focus on analyzing differences related to data types and to pre- and post-conditions in component interfaces. In [5] CORBA IDL is extended based on pi-calculus to incorporate protocol definition that is then used in checking for components compatibility and replaceability. In [13] the focus is on developing adapters for software components that have compatible functionality but incompatible interfaces and protocols. However, this approach assumes that interface mappings are provided. These efforts provide mechanisms that can be leveraged for Web service protocols adaptation, but are not sufficient. In fact, service protocols require richer description models than component interfaces and protocols. This is because clients and services are typically developed by separate teams, possibly even by different companies, and service descriptions are all that client developers have to understand to know how the service behaves.

Another related area is the work on semantic Web services [15–17]. The idea here is to empower the description of Web services with semantic information using ontologies. Two main approaches in this area are: describing services using ontology languages such as OWL-S[3] [15], and annotating service descriptions with semantics (e.g., like in WSDL-S[4])[16]. The main idea of the latter approach

[3] http://www.w3.org/Submission/OWL-S/
[4] http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html

is to use the same set of WS-* family of standards as opposed to proposing new standards for Web service descriptions in OWL-S, and relax the choice of ontologies to some other alternatives in addition to OWL. The use of semantics information in form of shared ontologies diminishes the semantic heterogeneities of service interfaces and protocols, thus simplifies service adaptation [16, 17]. However, the issues related to defining reference ontologies to be shared, ontology compatibility checking, merging and customization has affected the wide acceptance of these approaches.

In a nutshell, in this work we aim at transforming the development of adapters for web services from a tedious and error prone work to a process that is model-driven and, to a large extent, automated. To the best of our knowledge, there is no existing work that considers this problem for Web services.

References

1. Alonso, G., Casati, F. , Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures, and Applications. Springer, (2004)
2. Ryan N. D., Wolf, A. L.: Using Event-Based Translation to Support Dynamic Protocol Evolution. ICSE'04. Edinburgh, Scotland, United Kingdom (2004)
3. Benatallah, B., Casati, F., Grigori, D., M. Nezhad, H.R., Toumani, F.: Developing Adapters for Web services Integration. CAiSE, Porto, Portugal (2005)
4. Benatallah, B., Casati, F., Toumani, F.: Representing, Analysing and Managing Web Service Protocols. Data and Knowledge Engineering. Elsevier Science (2005)
5. Canal, C., Fuentes, L., Pimentel, E., Troya, J., Vallecillo, A: Adding Roles to CORBA Objects. IEEE TSE, 29(3), (2003)
6. Dong, X., Halevy, A. Y., Madhavan, J., Nemes, E., Zhang, J: Similarity Search for Web Services. VLDB Conference. Toronto, Canada, (2004)
7. Ponnekanti, S. R., Fox, A.: Interoperability among Independently Evolving Web Services. Middleware. Toronto, Canada (2004)
8. Bordeaux et al: When are two Web Services Compatible? VLDB TES, (2004)
9. Wang, Y., Stroulia, E.: Flexible Interface Matching for Web-Service Discovery, WISE, Roma, Italy, December (2003)
10. Baina, K., Benatallah B., Casati, F., and Toumani, F.: Model-Driven Web Service Development. CAiSE, Riga, Latvia, June (2004)
11. Wombacher, A., Mahleko, B., Fankhauser, P., Neuhold, E.: Matchmaking for Business Processes based on Choreographies. EEE. Taipei, Taiwan (2004)
12. Rahm, E., Bernstein, Ph. A.: A Survey of Approaches to Automatic Schema Matching. VLDB J. 10(4), (2001) 334-350.
13. Yellin, D. M., Strom, R. E.: Protocol Specification and Component Adaptors. ACM TOPLAS, 19(2), (1997)
14. Zaremski, A. M., Wing, J. M.: Specification Matching of Software Components. ACM TOSEM, 6 (4) , October (1997)
15. Martin, D., et. al.: Bringing Semantics to Web Services: The OWL-S Approach. SWSWPC, San Diego, USA (2004)
16. Patil, A.A., Oundhakar, S.A., Sheth A.P., Verma, K.: Meteor-s Web Service Annotation Framework. WWW '04, New York, USA (2004)
17. Williams, S.K.; Battle, S.A.; Cuadrado, J.E.: Protocol Mediation for Adaptation in Semantic Web Services. Hewlett-Packard Technical Report (HPL-78). May (2005)

Towards A Transaction Framework for Contract–Driven, Service–Oriented Business Processes

Ting Wang

Information Systems Subdepartment, Department of Technology Management,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
t.wang@tm.tue.nl

Abstract. Transaction support is vital for reliability of business processes which nowadays can involve dynamically composed services across the Internet. However, no single transaction model is comprehensive enough to accommodate various transactional properties demanded by those processes. Therefore we intend to develop a Business Transaction Framework (BTF) that utilizes existing ingredients in the context of contractual business relations. The proposed BTF has flexibility by using a library containing abstracted transaction models, validity based on formalization and business trustworthiness guaranteed by contractual specifications of transactional qualities.

1 Introduction

E-business has moved from the simple automation of business applications to the complex integration and coordination of business processes that belong to different organizations. Business processes have grown to be very intricate to involve miscellaneous activities and resources, and often need to invoke services provided by others. In such an environment, various business relationships need to be regulated, which result in a large amount of contracts along the processes. In addition, the dynamically composed activities usually have complex interrelated dependencies which make exceptions and errors prone to occur during the process execution. Suppose there is a booking process in a travel agency which needs to invoke three Web services, hotel booking, car rental and flight booking respectively. Besides there are activities like billing, payment check etc. which happen internally. These activities exhibit various transactional semantics and the process in a whole may demand different transactional qualities during execution. For instance, when a customer cancels the booking before payment, the previous completed activities have to be rolled back as if the process has never been initiated. However in case of a cancel after payment, the process probably can not be returned to the exact same state when it started. Therefore the process needs comprehensive and flexible transactional support to guarantee reliability.

To address the above concern, the XTC (eXecution of Transactional Contracted electronic services) project was proposed which aims at laying a generic

foundation to the transactional support for processes in service-oriented environment. Within this project, my PhD research focuses on the development of a Business Transaction Framework (BTF) for contract-driven, inter-organizational business processes and in particular, on contractual agreements specifying transactional qualities of those business processes.

The rest of the paper is organized as follows. We describe the research background that motivates our work and review the related work in Sect.2. We introduce the BTF and discuss our research approach in Sect.3. We end this paper in Sect.4 with the summary and the discussion of future work.

2 Research background and related work

The challenge of e-business nowadays is to integrate distributed applications into a corporate process which enables free and secure data-flow and control-flow across the organizational boundaries. The former way of tightly coupled integration like EAI (Enterprize Application Integration) can no longer meet today's requirements. To loosely couple the applications, one approach is wrapping them as ready-to-use services. The paradigm of Service-Oriented Computing (SOC) and its underlying Service-Oriented Architecture (SOA) [1, 2] allow distributed services ready to be utilized by application developers despite their locations, platforms and internal implementations. While the services may refer to any piece of functional software over the network (the Internet, Intranet or the Grid), Web services have been considered by the academia and the industry a fundamental technology to implement SOA [2–6]. Having moved beyond the basic layer of the extended SOA [2] focusing on service publishing, discovering and binding, current endeavors of Web services are made towards the higher layers of service composition and management.

Transaction management in Web services-based SOC paradigm is an intriguing topic in the research of service composition and management. First appeared in the mid 1970s to address the multi-user, concurrent-program concern in databases, transaction management has become a prevalent mechanism to guarantee reliability to running applications. Earlier work like [7–10] on ACID transactions and advanced transaction models is more or less database-oriented thus not suitable for complex business processes. Later on, research (e.g. [11, 12]) flourished to address the transactional support for business process automation and integration. Meanwhile, driven by the growing need of service composition, Web services transaction specifications are proposed by different standardization bodies (e.g. [13]). However, the above mentioned process-oriented transactions are either predefined in structure before process execution (e.g. workflow transactions) or not suitable for other types of applications (e.g. Web service transactions). A possible way to realize transaction support for a complex business process, which usually consists more than one Web services, is to orchestrate loosely coupled services into a single business transaction by guaranteeing coordinated, predictable outcomes for the participating partners [14]. Therefore we target the development of a high-level transaction framework that is more

flexible and comprehensive to accommodate various transactional semantics and applications.

Another field of our interest is e-contracting, which ensures business processes with legal liability and cost-efficiency. To sustain a certain level of process reliability, we propose to use contracts as service level agreements to specify the agreed transactional qualities of the whole business process as well as each component service. For example, we may leverage the WSLA (Web Service Level Agreement) framework [15] to enclose transactional semantics in the SLAs that serve as contracts to bond the service providers and consumers together. Work on the general Quality of Service (QoS) for processes such as [16] can serve as the starting point. We suppose that the desired Transactional Quality of Service (Tx-QoX) is a subset of the QoS for Web services composition. For instance, cost is a very common non-functional Web service QoS but will be excluded in our case. Also work on contractual support for Web services like [17, 18] can serve as a basis for us to develop a contractual approach to specify Tx-QoS. With regard to the specification language for such contracts, we consider e-contracting languages and the WSLA language as candidates. Currently there are some proposals about e-contracting languages such as [19, 20], but none is widely adopted to our best knowledge. The WSLA language may well define normal Web services QoS but seems too constraint for the expression of transactional semantics. Therefore we intend to develop a specification language for our purpose of contractual support for Tx-QoS.

Today's business environment, which involves multiple partners, diverse processes and heterogeneous systems, motivates us to develop a BTF with its flexibility by using the ATC (Abstract Transactional Construct) library, validity based on BTF/ATC formalization and business trustworthiness guaranteed by the contractual specification of Tx-QoS.

3 Business Transaction Framework (BTF)

So far we have elaborated a conceptual framework with an underlying architecture that is shown in Fig.1. The basic idea is to utilize existing transactional models and abstract them into ATCs so that they can be selected and composed on demand. The ATC Editor creates, deletes and modifies ATCs through the ATC Library Manager in the definition phase. Fed by a process specification, the ACBT (Abstract Composite Business Transaction) is created by the ACBT Composer and stored in the ACBT library in the composition phase and leverages ATCs as its building blocks. During the execution phase, the CBTs (Composite Business Transaction) are instantiated from the ACBT by the CBT Creator and consist of TCs(Transaction Constructs) instantiated from ATCs. Each CBT Manager monitors and manages one CBT. At the highest level there is a BTF Manager that coordinates the activities of other function components and manages transactions through all three phases of the process life cycle. x

To ensure a flexible and valid transaction framework, we employ classic and widely adopted transaction models and abstract them to shape the ATC library.

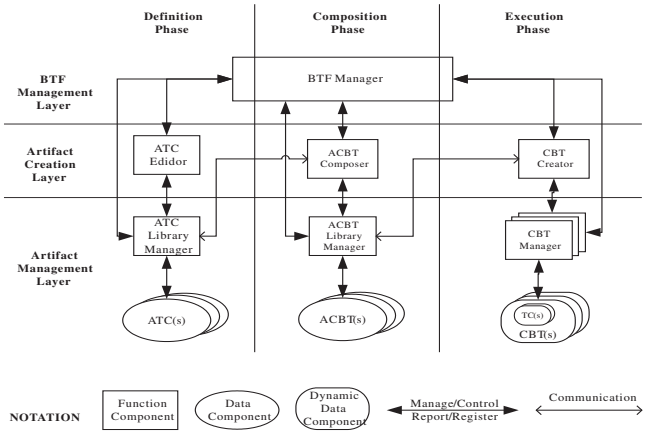

Fig. 1. BTF Architecture

For each ATC, we define a set of parameters, among which some are assigned in the definition phase to specify its internal structure while the others are assigned in the composition phase to specify its position in an ACBT. We design the framework based on the Web services standards and place it on top of the composition layer. To implement the BTF, we develop a reference architecture delivering the necessary functionalities by following the rules and techniques suggested in [21], with the focus on the qualities like modifiability, reusability and portability. This architecture serves as a blueprint to design a prototype system (e.g. on WebSphere platform) running some business processes based on real cases.

One advantage of the proposed BTF lies in the quality control by means of contractual agreements between the service providers and consumers. So far there is no available research on contractual transaction specifications. Besides, there is no clear definition about what are the measurable transactional qualities of business processes. As a business process may contain activities exhibiting different transactional semantics, the supporting transaction framework should accommodate the required Tx-QoS for each component activity as well as the whole process flexibly. Again we take the travel booking process of Sect.1 for example, which includes multiple parties like a customer, a travel agency and candidate hotels, airlines and car-rental companies. Each company involved in this process has its own internal automated processes and offers business functions through Web services (here we assume each component activity is automated and wrapped as a Web service). The whole process can be viewed as a hybrid transaction consisting of various sub-transactions, each dealing with a transactional Web service. It is very likely that these component services require different transactional qualities such as the unconventional payment atomicity [14] for the payment activity. Meanwhile the composite process demands a cer-

tain level of qualities for the whole transaction and the demand may vary over time. We propose to extend the contractual agreements (for example, the WSLA files) to specify the expected or agreed transactional properties for each component service. Derived from these agreements, we can specify the Tx-QoS clauses in the process contract. Therefore the trustworthiness is guaranteed by means of legal contracts.

4 Summary and future work

Our preliminary design of the conceptual and architectural BTF/ATC described in Sect.3 is the first step towards a comprehensive and flexible transaction framework for contract-driven, service-oriented business processes. The idea is to extract and group existing transaction models into an ATC library and pick up the needed ones to compose a transaction hierarchy on demand. The design work is far from complete, with several directions for the future elaboration:

The first direction is to achieve a complete and sound BTF/ATC specification. To guarantee the correctness of the proposed BTF, we need to develop a method to extract and connect ATCs into an ACBT. One approach is to encapsulate the transactional semantics with parameterizable interfaces based on an ATC taxonomy. This way an ACBT can be composed through these interfaces by predefined composition rules.

Secondly, a refined architecture with a complete set of components delivering proper functionalities is required as the blueprint to implement a prototype system. Bearing the unique features (e.g. flexibility, extensibility, compatibility) of the BTF in mind, we plan to decompose some of the function components shown in Fig.1 to get a deeper level of architectural overview. For example, the BTF manager can be decomposed into a second-level architecture that may contain a messenger, a contractor and a monitor and the contractor can be further refined if needed.

Last but not least, we are going to develop a Tx-QoS specification language. To achieve this, we need to investigate the existing e-contracting languages. In addition, a clear description of intended QoS for transactions is needed. Afterwards we expect to combine the contractual Tx-QoS with the BTF specification.

Acknowledgement

The XTC project is funded by the Dutch Organization for Scientific Research (NWO No. 612.063.305). Thanks go to Paul Grefen, Benedikt Kratz and Jochem Vonk whom I work with in this project.

References

1. Papazoglou, M.: Service -oriented computing: Concepts, characteristics and directions. In: Procs. of the 4th Int. Conf. on Web Information Systems Engineering (WIDE'03). (2003) 3–12

2. Papazoglou, M., Georgakopoulos, D.: Service-oriented computing. Comunications of the ACM **46** (2003) 25–28
3. Brown, A., Johnston, S., Kely, K.: Using service-oriented architecture and component-based development to build web service applications. White paper, Rational Software (2002)
4. Lehmann, M.: Deploying large-scale interoperable web services infrastructures. Web Services Journal **5** (2005)
5. Mahmoud, Q.H.: Service-oriented architecture (SOA) and web services: The road to Enterprise Application Integration (EAI). Technical article, Sun Developer Network (2005)
6. Ferguson, D.F., Lovering, B., Storey, T., Shewchuk, J.: Secure, reliable, transacted web services: architecture and composition. Technical article, Microsoft Corporation and IBM Corporation (2003)
7. Haerder, T., Reuter, A.: Principles of transaction-oriented database recovery. ACM Comput. Surv. **15** (1983) 287–317
8. Moss, J.E.B.: Nested transactions: an approach to reliable distributed computing. PhD thesis, EECS Department, M. I. T. (1981)
9. Garcia-Molina, H., Salem, K.: Sagas. In: Procs. of the 1987 ACM SIGMOD Int. Conf. on Management of data (SIGMOD'87). (1987) 249–259
10. Chrysanthis, P.K., Ramamritham, K.: ACTA: a framework for specifying and reasoning about transaction structure and behavior. In: Procs. of the 1990 ACM SIGMOD Int. Conf. on Management of data (SIGMOD'90). (1990) 194–203
11. Alonso, G., Agrawal, D., Abbadi, A.E., Kamath, M., Gunthor, R., Mohan, C.: Advanced transaction models in workflow contexts. In: Procs. of the 12th Int. Conf. on Data Engineering (ICDE '96). (1996) 574–581
12. Grefen, P., Vonk, J., Apers, P.: Global transaction support for workflow management systems: from formal specification to practical implementation. The VLDB Journal **10** (2001) 316–333
13. Cabrera, L.F., et al.: Web Services Transactions. Available at http://www-128.ibm.com/developerworks/library/specification/ws-tx/ (2005) Including WS-Coordination, WS-AtomicTransaction and WS-BusinessActivity specifications.
14. Papazoglou, M.: Web services and business transactions. World Wide Web: Internet and Web Information Systems **6** (2003) 49–91
15. IBM Corp.: Web Service Level Agreements (WSLA) Project. Available at http://www.research.ibm.com/wsla/ (2004)
16. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of services for workflows and web service processes. J. of Web Semantics **1** (2004) 281–308
17. Tosic, V., Pagurek, B.: On comprehensive contractual descriptions of web services. In: Procs. of the 2005 IEEE Int. Conf. on e-Technology, e-Commerce and e-Service (EEE'05). (2005) 444–449
18. Milanovic, N.: Contract-based web service composition framework with correctness guarantees, The 2nd Int. Service Availability Symposium (2005)
19. Angelov, S., Grefen, P.: Requirements on a B2B e-contract language. Beta working paper WP 140, Eindhoven University of Technology (2005)
20. Berry, A., Milosevic, Z.: Extending choreography with business contract constraints. Int. J. Cooperative Inf. Syst. **14** (2005) 131–179
21. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Longman (1998)

A Business Aware Transaction Framework for Service Oriented Environments*

Benedikt Kratz

Tilburg University, Infolab
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
B.Kratz@uvt.nl

Abstract. Transaction support is a key ingredient for reliable inter-organizational business transactions between distributed, loosely coupled, heterogenous systems. In this paper we present the outline and basic concepts of the Business Transaction Framework research project. In this project we want to develop a flexible and integrated framework that supports the structured application of abstract transactional constructs on business transactions. The ultimate goal of the project is the creation of reliable business transactions in the context of service oriented environments.

1 Introduction & Background

Growing economic activity and technical developments have lead to ever increasing interactions between businesses, thereby necessitating the coupling of business processes. The internal business processes of organizations need to be aligned with the external interactions in order to allow an efficient and effective realization of business goals. In this context, a business transaction occurs when the business processes of two (or more) organizations are connected and information is exchanged that changes the internal state of the organizations (we abstract here from the flow of physical goods). The consistent execution of business processes and registration of occurring business transactions is of utmost importance for organizations as these determine the profitability of an organization. With the emergence of information systems that support business processes and business transactions, automatic execution and registration of these processes and business transactions is now possible.

Transaction support is a well-known mechanism to ensure consistency in a multitude of situations. Using transactions to make changes on (distributed) data to reflect real world state changes (the state of the data reflects the state of the outside world as modeled by the relationship between them) has been a mature application within the database realm for multiple decades now. Transactions are guided by transaction models, like the ACID model, that define the significant

* The research reported in this paper is part of the eXecution of Transactional Contracted Electronic Services (XTC) project (No. 612.063.305) funded by the Dutch Organization for Scientific Research (NWO).

events that pertain to transactions adhering to that model [1]. The transaction concept is also applied within applications as application level transactions, e.g., within Enterprise JavaBeans. Transaction support for workflows has also been high on the research agenda over the past few years (e.g., [2]).

In comparison with the above transactions, business transactions that occur in a business process are quite different. Business transactions are complex and multi-level, span many organizations, involve multiple parties with heterogenous systems, usually have a long duration and optional parts and are automatically executed. When developing transaction support for business transactions, these characteristics need to be addressed to ensure a consistent execution of business transactions. Therefore, the business driven business transactions and technology driven data and application level transactions and supporting transaction models must be aligned.

In this paper, we outline the Business Transaction Framework (BTF), which is the core element of our research project. The goal of our research project is the development of concepts, constructs and mechanisms as well as infrastructure to enable flexible transactions in the context of e-business. To achieve our research goal we focus on the following points:

- Specification of the BTF, in particular the required concepts and supporting architecture.
- Representation of business semantics in transaction support for business transactions.
- Emulation of other existing similar transaction frameworks within the BTF.
- Formalization of the above points.

Our focus here is solely on transaction support, we do not for example focus on security aspects as these are out of the scope of the research project.

The interest into business transactions has recently gained momentum as the Service Oriented Computing (SOC) paradigm recognizes the need for business transactions and emphasizes their importance in loosely coupled and dynamic service oriented environments. The BTF is connected to the SOC paradigm in two ways. First, as SOC is the enabler of interoperability between heterogenous systems accompanied with loose coupling and highly dynamic establishment of business relations, we leverage the SOC philosophy to implement the BTF with Web Service technology. Secondly, the BTF extends SOC transaction models to address business driven requirements (intra- and inter-organizational) on automated long running business transactions. Therefore we abstract from concepts (e.g., context), also allowing the integration of related concepts from fields like databases and workflows to achieve an integrated approach in heterogeneous environments.

To present our research project, the remainder of this paper is organized as follows. In Sect. 2, we present the motivation for this research together with an exploration of related work. We point out several issues that are not addressed by current research. In Sect. 3, we present a short overview of the BTF concepts together with a short illustration. The paper ends in Sect. 4 with a summary and an outlook of future work.

2 Motivation & Related Work

In the area of business transaction automation there are a variety of issues, both from a technical and a business perspective, that need to be addressed and that need to be aligned before an automated execution of business transactions and processes with web service technology becomes feasible.

Starting with the technical perspective, it is obvious that a business process with long running business transactions requires a broad support of applicable transaction models to allow a consistent execution. First of all, the (sub) process structure (e.g., nesting, routing, etc) limits the suitable transaction models as transaction models usually support only a particular structure (e.g., the open nested transaction model supports nesting). Also the participants of a business process require particular transaction models as these participants can be heterogenous (e.g., databases, workflow engines or web services) in nature.

Current approaches (e.g., [2–5]) to address these above technical issues however only use one or a limited combination of transaction models, applicable only in particular environments (e.g., databases, workflows, web services) in an ad hoc manner. The focus of these transaction models (e.g., sagas, open nested, etc.) is on application level transactions and not on business level transactions. Other research into transaction support for workflows, e.g., [2], business processes and web services, i.e., WS-Tx [4, 5] and WS-CAF [6], is geared towards the use of a restricted context and domain specific set of transaction models. The combination of transaction models is usually fixed, while there is a need for support of any (suitable and feasible) combination of transaction models.

Several approaches to integrate advanced transaction models exist. The ACTA framework [1] facilitates the formal specification, analysis and composition of advanced database transaction models. In [7], a meta-model for web service transaction models is proposed. The approach in [7] aims at flexible transaction support for web services using existing transaction models. However, the sole focus is on the modeling and representation of transaction models (selecting or creating one transaction model per application as opposed to the composition of different transaction models as required for complex sets of business transactions), without any execution framework with a detailed architecture and a clear coupling between business process and transactions. Both, [1] and [7], are limited in their applicability because of their from database technology driven development without taking into account business and workflow requirements.

Apart from the incomplete satisfaction of technical issues, business issues are not addressed by current transaction models and only very limited by current research. When looking at a business process from a business perspective, transaction support is required for the complete business protocol that defines the public, agreed business interactions between business parties. The interactions involve negotiations, commitments, contracts, shipping and logistics, tracking, varied payment instruments and exception handling. In addition to these concepts, the phase a business process is in as well as the current level of the business process determine the required transaction support. In [8], a phased model is introduced that distinguishes between pre-transaction, main transac-

tion and post-transaction phases in a collaborative business process. Because of aggregation and abstraction requirements in inter-organizational settings, [9] identified the need for a three level (external, interface and internal level) process framework to enable clear process descriptions in e-business system integration.

There is also a need for a more well-founded and formal approach towards the use of traditional transaction processing concerns of e-business. In [10], several atomicity criteria for business to consumer micro transactions that play a role in e-commerce are defined. The goal of our research is to use the atomicity criteria (like payment or contract atomicity) in large-scale e-business interactions as these provide business semantics at the business level to business transactions as opposed to the current, above mentioned, transaction models. A clear interrelation of these high level criteria with lower-lever transaction models needs to be established. This also enables the creation of a precise model that could help business partners in conflict situations (e.g., if a trading partner supports different atomicity criteria than another business partner).

3 The Business Transaction Framework

The BTF currently provides an initial conceptual model that will be briefly presented in this section. Currently, the focus of the framework is on the abstraction and composition of transaction models to address the technical issues as mentioned in Sect. 2. At present, we focus on the transaction support of business transactions from one participants viewpoint (i.e., orchestration oriented), and related to that, their business processes that contain these business transactions.

The conceptual model of the BTF describes and relates the concepts used in the framework. The framework's main concepts are distributed across three distinct phases (i.e., definition, composition and execution) to address the logical workflow of adding transaction support to business transactions. The need for a broad range of transaction models in a business conversation makes it necessary to define a transaction plan. The transaction plan connects different transaction models with each other to reliably execute the business process. The transaction models are the building blocks of a transaction plan, which in fact is a composition of transaction models. The definition of the building blocks is situated in the first phase of the framework. The second phase is concerned with the composition of transaction models from the first phase into the transaction plans. To be able to compose transaction models, existing transaction models with their properties and behavior must be analyzed and then it must be investigated what and how transaction models can be connected together and what the semantics of such a connection is. These first two phases are design-time phases. The compositions of transaction models can then be executed during the final, run-time (i.e., execution) phase of the framework.

The concept used in the first phase are the atomic building blocks of the framework called Abstract Transaction Constructs (ATC). An ATC is an abstract representation of a transaction model with its specific properties and functionality comparable to a transaction model class in object orientation. ATCs

are (if allowed by the particular transaction model) composable horizontally, allowing to create a choreographed ordering of ATCs, and vertically, allowing to nest ATCs. The behavior of an ATC is parameterizable through its structural (i.e., the process structure) and behavioral (i.e., the behavior during run-time) interfaces. To investigate and achieve interoperability between and reusability of transaction models we propose a taxonomy of transaction models in which we initially group existing transaction models by using the abstraction relation into three categories, i.e., flat (e.g., WS-AT [4]), choreographed (e.g., LRAs [6]) and nested (e.g., WS-BA [5]) transaction models.

Injecting transactional capabilities on business processes requires as inputs the business process specification, the collection of ATCs, ATC composition rules and contracts constraining the other inputs. When composing the ATCs together by instantiating their structural interface, Abstract Composite Business Transactions (ACBT) are created. An ACBT is a structural transaction specification for a business process comparable to a workflow specification.

If a business process is started, the accompanying ACBT is instantiated by parameterizing the behaviorial interface of the ATCs connected in the ACBT. The result will be a Composite Business Transaction (CBT) with fully parameterized ATCs called Transactional Constructs (TC). TCs are the concrete transactions executed at some point during the business process.

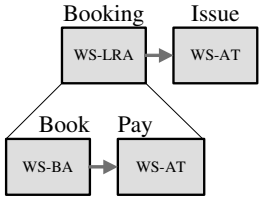

Fig. 1. ACBT example

In Fig. 1, a small example of an ACBT is presented. This ACBT is based on a simple (part of a) business process in which a travel can be arranged. The travel arrangement process contains one activity (*Booking*) with two subprocess steps and thereafter one further activity. Within the *Booking* activity, the subtasks of the *Book* subprocess (booking a hotel and a flight, both business transactions with other organizations; not shown) can be performed in parallel, after which the selected travel package is invoiced and the payment is registered and the tickets are electronically issued. Based on some process requirements for transaction support, ATCs can be composed into the presented ACBT. The WS-CAF LRA ATC is based on the SAGA transaction model and can be used to split the long-running *Booking* activity into smaller, shorter-running process steps, using compensation steps in case (part of) the process needs to rollback. A nested ATC (i.e., WS-BA) would suit the *Book* subprocess. A flat transaction model ATC (WS-AT) is used for the payment and issuing of the tickets. If the last activity (*Issue*) fails, it should be redone until it succeeds (forward recovery). Failing this activity should have no further impact on the rest of the already completed activities. Also, the payment task can be supported by a flat transaction model as its compensation is taken care of by the (higher level) WS-LRA.

4 Summary & Future Work

We explored the requirements for a business aware transaction framework (BTF) in service oriented environments. We introduced the concepts of the BTF to remedy current limitations of transaction support for business processes with business transactions that require an automated and consistent execution with clear semantics (in case of errors) in service oriented environments. Current approaches are not able to provide a platform-independent way of integrating required transaction models accommodating the various activities and their resources (databases, workflows, web services) throughout the business process into a cohesive whole. Representing transaction models by Abstract Transactional Constructs and composing these into Abstract Composite Business Transactions allows the flexible creation of reusable transaction plans to support business processes. The execution behavior of ACBTs is parameterizable at run-time in order to create Composite Business Transactions.

Future work for the BTF from a technical viewpoint will concentrate on the precise definition of the semantics of interactions between the transaction models. This will result in a set of composition rules for ATCs. Formal analysis is required to define the correctness criteria of resulting compositions. To support the BTF we are also currently developing a dynamic and flexible three-phased and three-level architecture. Once the technical requirements for the BTF are fulfilled, the business requirements will be addressed by incorporating unconventional atomicity criteria into ATCs. Next to the incorporation of existing standardization efforts from the business process arena (e.g., RosettaNet, EbXML), the BTF will make a shift from an orchestration oriented approach to a choreographed approach. We also plan to develop a proof of concept prototype.

References

1. Chrysanthis, P.K., et al: Synthesis of extended transaction models using acta. ACM Transactions on Database Systems **19** (1994) 450–491
2. Grefen, P., et al: Database Support for Workflow Management: The WIDE Project. Kluwer Academic Publishers (1999)
3. Elmagarmid, A.K., ed.: Database Transaction Models for Advanced Applications. Morgan Kaufmann (1992)
4. Cabrera, L.F., et al: WS Atomic Transaction (WS-AT). (2004)
5. Cabrera, L.F., et al: WS Business Activity Framework (WS-BA). (2004)
6. Bunting, D., et al: WS Transaction Management (WS-TXM). (2003)
7. Hrastnik, P., et al: Using advanced transaction meta-models for creating transaction-aware web service environments. International Journal of Web Information Systems **1** (2005)
8. Papazoglou, M.: Web services and business transactions. World Wilde Web: Internet and Web Information Systems **6** (2003) 49–91
9. Grefen, P., et al: A framework for e-services: a three-level approach towards process and data management. IBM Research Rep. RC22378, IBM Res. Division (2002)
10. Tygar, J.D.: Atomicity in electronic commerce. Proc. of the 15th annual ACM symposium on principles of distributed computing, ACM Press (1996) 8–26

Modeling and Analysing Web Services Protocols

Julien Ponge

ponge@isima.fr – http://www.isima.fr/ponge/
Laboratoire LIMOS, ISIMA – Campus des Cézeaux
63173 Aubière cedex, France

Abstract. Web services technology is emerging as the main pillar of service-oriented architectures (SOA). This technology facilitates application integration by enabling programmatic access to applications through standard, XML-based languages and protocols. While much progress has been made toward providing basic interoperability among applications, there are still many needs and unexploited opportunities in this area. In particular, services in SOAs require richer description models than object or component interfaces. This is due to the loose coupling inherent in SOAs and therefore to the fact that services are developed independently of clients. Hence, service descriptions need to include all the information needed by clients to understand if they can interact with a service and how. We outline in this paper a novel approach developed as part of a PhD research work, based on web services protocols descriptions that allows for a high-level analysis based on operators.

1 Introduction

There is still a lot to be done to simplify service development and interaction. In particular, an important aspect of Web services that affects interoperability is that services are loosely-coupled, that is, are not developed only to interact with specific clients but are meant to serve the needs of many different clients, possibly developed by different teams or even different companies. Hence, developers of client applications need to be aware of all functional and non-functional aspects of a service to be able to understand if they can/need inter-operate with a service and how to develop clients that can interact correctly with the service. For this reason, service descriptions should be richer than ”just” descriptions of interfaces as in conventional middleware. Specifically, it is commonly accepted that a service description should include not only the interface, but also the *business protocol* supported by the service, i.e., the specification of possible message exchange sequences (conversations) that are supported by the service as well as other useful abstractions (temporal constraints, transactions, vendor policies and so on).

The PhD research work outlined in this paper aims at developing a novel solution to simplify the web services life-cycle management by providing a framework for modeling and analysing web services described as extended protocols. This work is part of a larger effort materialized by the ServiceMozaic platform [1] that enables to support design, development and management of web services in a model-driven CASE tools set environment. The paper first outlines this general framework, then reviews the state of

the art in the field of web services modeling and management. We then expose what we have done so far by considering the temporal constraints abstraction to perform services compatibility and replace-ability analysis. Finally, we conclude and provide directions for future work.

2 Toward a model-driven framework for web services

Protocols specifications describe the external behaviors of services, making them essential for developers willing to create clients that can interact correctly with services. We argue that protocols specifications can considerably simplify web services life-cycle management. For example, during web services development, protocols of clients and providers (possibly obtained from normative efforts such as RosettaNet[1]) can be analyzed to identify which conversations can be carried out between two services, therefore reducing potential runtime errors and suggesting possible modifications to improve the compatibility with the service. Protocol analysis and management can also provide valuable help to support *change support and evolution*. Indeed, it could enable the eased identification of the modifications required by clients at the protocol level when services protocols change. There are also promising applications like automated exceptions handling, compliance verification, and static or dynamic binding, all by taking advantage of the protocols descriptions. To realize such benefits, we aim at developing a conceptual framework, and the associated CASE tool environment, named ServiceMozaic, that enables to support design, development and management of web services. Our primary goal is to develop a model-driven framework within which whole, or at least, central parts of web services are generated and managed from models. The related research issues are the following.

- *Protocol modeling.* We built our framework upon an extended protocol model with formal semantics that allows richer services descriptions. Beside its ability to describe messages choreography constraints (i.e., the legal messages exchange sequences), the proposed model includes relevant abstractions such as temporal constraints, that enables users to better understand the external behavior of services. To avoid creating a model that would be either too simplistic, or too complex, we developed it upon an analysis of real-world e-commerce portals in order to identify the abstractions that can be indeed useful for practitioners.
- *Protocol analysis and management.* We target three types of protocols analysis, namely *compatibility*, *replace-ability* and *consistency* analysis. Compatibility analysis consists in checking whether two services can interact correctly, based on their protocol definitions (i.e., whether a conversation can take place between the considered services). In turn, replace-ability analysis refers to the verification of whether two protocols can support the same set of conversations (e.g., a service can replace another one in general, or when interacting with specific clients). Finally, consistency analysis is related to supporting changes in protocols and their impact on requesters but has not been further investigated.

[1] See `http://www.rosettanet.org/`

- *Protocol algebra and protocol management operators.* A distinctive feature of our approach lies in the definition of operators to query, analyze, and transform protocols. Such operators are the key to carry out most of the features described above.
- *Protocols adapters and code generation.* Theses two issues are part of the Service-Mozaic platform but won't be further detailed in this paper. Briefly, adapters [2] can be created when compatibility (resp. replace-ability) analysis between two protocols reveals that there exist some mismatches that make them not fully compatible (resp. replaceable). Adapters allow to enhance the compatibility / replace-ability level in such situations.

3 State of the art

Tools supporting web services development today are mainly concerned with interoperability issues at the lower levels of the web services stack, like the mappings from WSDL descriptions to Java/C# source code and vice-versa, making two SOAP-based systems talk to each other. Similarly, the existing standards in the higher-level services descriptions such as BPEL4WS, WSCI or WSCL proved to be more concerned with implementation aspects than enabling the kind of formal analysis that we envision. Indeed, the importance of formal analysis of web services protocols in terms of automated support for services interoperability at the business protocol level has been discussed in recent papers: [1, 3–5]. Several efforts recognize aspects of protocol specification in component-based models [6, 7]. They provide models (e.g., pi-calculus based languages for component interface specifications) and algorithms (e.g., compatibility checking) that can be generalized for use in web services protocol specifications and management. Indeed, various efforts in the general area of formalizing web services description and composition languages emerged recently [5, 8]. However, in terms of managing the web services development life-cycle, technology is still in the early stages. Consistency analysis should have some interesting links with existing software engineering techniques such as refactoring [9].

Our approach, based on a protocol algebra and protocol operators is novel in the field. Specifically, it should be more fine-grained than the approaches mentioned above when doing compatibility and replace-ability analysis. We believe that to some extent, the development of a protocol algebra for formal services descriptions analysis implemented inside a larger CASE tool, based on the identification of abstractions needed by practitioners [10], can have the same impact that relational algebra had for relational databases.

4 Web services protocols modeling and analysis

4.1 A model extended with temporal constraints

Our model is based on the *web services business protocol* proposed in [10, 1], which is built upon the traditional state machine formalism to represent messages choreography constraints. States represent the different phases that a service may go through during its interaction with a requester to the provider and vice-versa. A message corresponds

to the invocation of a service operation or to its reply. Hence, each state identifies a set of outgoing transitions, and therefore a set of possible messages that can be sent or received. Each transition is labeled with a message name followed by the message polarity, that is, whether the message is incoming or outgoing. The protocols are deterministic, that is to say, they have one initial state and, each state cannot provide more than one outgoing transition labeled with the same message. The model also supports the notion of *final states*, which correspond to the end of a successful conversation, in the sense that the messages exchanges between the provider and the requester is over on both sides. Briefly, the reason for using state machines as the basis for the model is because it is familiar to users, it is suitable to described reactive behaviors, and the notion of states is useful to perform services execution monitoring.

We have extended the model (called *timed web service business protocol*) to cater for temporal abstractions in [11, 12]. Transitions can become *timed transitions* when carrying temporal constraints. We identified two kinds of timed transitions. The first one (which we called *C-Invoke constraints*) relates to a time window during which the related operation can be triggered explicitly by either the provider or the requester. The other kind of transition that we identified corresponds to *timed implicit transitions* that can automatically occur once a certain date and time has been reached. We called this type of constraints *M-Invoke*. A proper discussion on the formal semantics of the model would require too much space to fit in this paper, but we can recall that they are based on the notion of *timed execution traces*, inspired by timed automata [13].

4.2 Temporal compatibility and replace-ability analysis

Reactive systems have been widely studied, notably to the benefit of software and hardware verification. For analysis purposes, two formal frameworks are of interest. The first one is the temporal logics theory [14]. It allows for expressing complex requirements expressed as formulas that have to be satisfied on reactive systems modeled as automata. For example, a complex safety property (*"the event will never happen"*) or a liveness property (*"the event will happen"*) can be described within this framework. It appears that we won't need temporal logics and their timed extensions (such as [15]) as we need to define simple timed constraints. The other framework is the timed automaton [13] which provides a lot of interesting automata classes, notably the event-recording automaton that has some valuable decidability and complexity properties on verification techniques. We have developed mappings from/to timed web services business protocols that are useful from a formal point of view. However, we try not to use straight the existing timed automata verification techniques which have been developed with the general cases in mind and are thus more demanding than the ad-hoc algorithms that we can develop for timed protocols. What's more, traditional verification techniques lack the identification of partial compatibility and replace-ability, which is an important contribution of this work. We have defined *classes* to identify different levels of compatibility and replace-ability, as well as *operators* that can be applied to protocols definitions to assess the level of compatibility and replace-ability. These classes can be characterized by using and combining 3 operators (the *timed compatible composition*, the *timed intersection* and the *timed difference*), for which we have a more detailed def-

inition as well as polynomial-time algorithms in [11]. We introduce the compatibility and replace-ability classes below.

- *Partial compatibility* (or simply, compatibility): A protocol P_1 is partially compatible with another protocol P_2 if there are some executions of P_1 that can inter-operate with P_2, i.e., if there is at least one possible conversation that can take place among two services supporting these protocols
- *Full compatibility*: a protocol P_1 is fully compatible with another protocol P_2 if all the executions of P_1 can inter-operate with P_2, i.e., any conversation that can be generated by P_1 is understood by P_2.
- *Protocol equivalence w.r.t. replace-ability*: two business protocols P_1 and P_2 are equivalently replaceable if they can be interchangeably used in any context and the change is transparent to clients.
- *Protocol subsumption w.r.t. replace-ability*: a protocol P_2 is subsumed by another protocol P_1 w.r.t. replace-ability if P_1 supports at least all the conversations that P_2 supports. In this case, protocol P_1 can be transparently used instead of P_2 but the opposite is not necessarily true.
- *Protocol replace-ability with respect to a client protocol*: A protocol P_1 can replace another protocol P_2 with respect to a client protocol Pc if P_1 behaves as P_2 when interacting with a specific client protocol Pc.
- *Protocol replace-ability with respect to an interaction role*: Let P_R be a business protocol. A protocol P_1 can replace another protocol P_2 with respect to a role P_R if P_1 behaves as P_2 when P_2 behaves as P_R. This replace-ability class allows to identify executions of a protocol P_2 that can be replaced by protocol P_1 even when P_1 and P_2 are not comparable with respect to any of the previous replace-ability classes.
- *Partial protocol replace-ability*: Partial replace-ability is when there is replace-ability but only for some conversations and not others. For example, we have partial replace-ability with respect to a client protocol when protocol P_1 can replace another protocol P_2 in at least some of the conversations that can occur with another P_C protocol.

5 Conclusion

After one year of work, we have proposed an extension of the web services business protocols model proposed in [1] to cater with temporal abstractions in [12, 11]. We proposed a novel approach by characterizing the temporal compatibility and replace-ability classes by using protocols operators for which we have efficient algorithms. We also have a link to the timed automata [13] theory that allows us to assess from a formal point of view that it has interesting properties. The implementation of the web services business protocol model has been done and will be soon extended with temporal abstractions. We have also implemented an editor for protocols in the ServiceMozaic platform as a GEF-based Eclipse plug-in[2].

Future work includes focusing on the third kind of protocols-based analysis: consistency analysis. We will also need to take into account multi-protocols choreography

[2] See `http://www.eclipse.org/` and `http://www.eclipse.org/gef/`.

analysis (the current work is focused on the analysis between a service provider and a requester). Finally, other useful abstractions such as transactional properties will provide another promising area of investigation.

References

1. Farouk Toumani, Boualem Benatallah, Fabio Casati: Analysis and Management of Web Services Protocols. DKE, Special issue from ER'04 (2004)
2. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In Springer-Verlag, ed.: Procs of CAiSE'05. Lecture Notes in Computer Science (2005)
3. Bordeaux, L., Salaun, G., Berardi, D., Mecella, M.: When are two Web Services Compatible? In: VLDB TES'04, Toronto, Canada. (2004)
4. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: a look behind the curtain. In: Proc. 22th Principles of Database Systems (PODS'03), San Diego, CA, USA, ACM (2003) 1–14
5. Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: a new approach to design and analysis of e-service composition. In: WWW 2003, Budapest, Hungary, ACM (2003) 403–410
6. Canal, C., Fuentes, L., Pimentel, E., Troya, J., Vallecillo, A.: Adding Roles to CORBA Objects. IEEE Trans. Software Eng **29** (2003) 242–260
7. Yellin, D., Storm, R.: Protocol Specifications and Component Adaptors. ACM Trans. Program. Lang. Syst. **19** (1997) 292–333
8. Mecella, M., Pernici, B., Craca, P.: Comatibility of e-services in a cooperative multi-platform environment. In: VLDB-TES'01, Springer (2001)
9. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring – Improving the Design of Existing Code. Addison Wesley (1999)
10. Benatallah, B., Casati, F., Toumani, F., Hamadi, R.: Conceptual Modeling of Web Service Conversations. In: Procs of CAiSE'03. Volume 2681 of LNCS., Klagenfurt, Austria, Springer (2003) 449–467
11. Boualem Benatallah, Fabio Casati, Julien Ponge, Farouk Toumani: Compatibility and replaceability analysis for timed web service protocols. In: Proceedings of BDA 2005, Saint-Malo, France. (2005)
12. Boualem Benatallah, Fabio Casati, Julien Ponge, Farouk Toumani: On Temporal Abstractions of Web Services Protocols. In: Proceedings of CAiSE Forum 2005, Porto, Portugal. (2005)
13. Rajeev Alur, David L. Dill: A theory of timed automata. Theoretical Computer Science (1994) 183–235
14. Pnueli, A.: The temporal logic of programs. In: FOCS. (1977)
15. Alur, R., Courcoubetis, C., Dill., D.: Model-checking in dense real-time. Information and Computation **1** (1993) 2 – 34

Web Service Discovery with Implicit QoS Filtering

Natallia Kokash

DIT - University of Trento, Via Sommarive, 14, 38050 Trento, Italy
email: natallia.kokash@dit.unitn.it

Abstract. Web Service (WS) discovery is a critical problem hindering web service technology proliferation. The current solution, based on catalog-style browsing, provides no control over the quality of registered services. Application of matching techniques for WS retrieval is still under investigation. The objective of this work is the design of a framework to improve WS discovery. Our approach is based on application of distributed recommendation system to provide Quality of Service (QoS) information and on testing of retrieval methods on service specifications.

1 Introduction

WS paradigm is a promising model of software technology, based on loosely coupled, distributed and independent services operating via the web infrastructure. To overcome platform and language dependence, services are described using Web Service Description Language (WSDL). Standardized XML-based interfaces help performing service reuse. Service descriptions are cataloged in Universal Discovery, Description and Integration (UDDI) registries. Although there exists a stack of standards to regulate the communication of processes and automated tools to convert legacy applications into web services, WS technology is still not widely used. One of the reasons is the lack of means to support WS discovery, i.e., the identification of existing WSs that can be used by new web applications. This problem is rather extensive and admits various interpretations [4]. Under automated discovery, a requester agent performs service search and evaluates the results. Currently UDDI registries are the dominating technological basis for WS discovery. They allow business compliance and reuse, and in perspective they could provide control over data and facilitate WS lifecycle management. But existing registries are still small and mostly private, there is no control over provided information, qualitative characteristics of WS and ability of quality-based retrieval. The discovery supported by UDDI API is inaccurate as services retrieved may be inadequate due to low precision and low recall. Such mentioned disadvantages determine our objective: we propose a framework for efficient WS discovery that provides clients with QoS information and reduces the probability of failure by analyzing statistics of previous service invocations.

The rest of the paper is organized as follows. In Section 2, we discuss different aspects of the problem and present the existing work relevant for our research.

Section 3 describes our approach. In Section 4, we draw some conclusions and outline future work.

2 Background

We can judge how well a web service satisfies a client's goal using various types of matching. *Signature matching* considers only function types without regarding their behavior. *Specification matching* is a way to compare two software components, based on descriptions of their behaviors. Components can be compared with various degrees of accuracy (exact and relaxed matching). However, this approach requires formal pre/postcondition specifications. Hausmann et al. [5] examine the application of such methods for WS discovery. WS matching is related to the automatic schema matching which is a basic problem in many application domains like data integration or semantic query processing. Rahm et al. [13] provide a good survey in this area. In *syntactic matching* we look for the similarities into data using syntax driven techniques. In *semantic matching* the key intuition is the mapping of meanings. For example, *surname*, *family name*, *cognomen* and *last name* represent the same concept. One of the relevant proposals in the field of WS discovery is to use an extension of UDDI that contains WSDL specifications [7]. In this way, dynamic retrieval through common terminology and shared meaning is enabled. WSDL does not provide any special semantic specifications but it contains the `<documentation>` element with service documentation and elements with natural language descriptions of operations and data types. Identifiers of messages, operations and data types are meaningful, and XML syntax allows to capture the domain specific relations. *Semantic Web services*, i.e., web services empowered with formal ontologies [16], revealed new perspectives in service automatic discovery, composition and execution monitoring (e.g., [10]). Since this solution is based on predefined ontologies, it is not flexible and contradicts the dynamic nature of web-based interaction.

From the WS discovery perspective, the major criterion of WS retrieval system evaluation is the relevance of found services with respect to the end user. The most popular quality evaluation measures of Information Retrieval (IR) engines are *precision* (a measure of the usefulness) and *recall* (a measure of the completeness). They have a fixed range and are easy to compare across different queries and engines, but they do not account for the quality of ranking. For a machine learning algorithm we need an effective single number measure. An *average precision* that combines precision, relevance ranking, and overall recall is an ideal measure for our task where good ranking is extremely relevant due to invocation cost of the services.

Assuring the WS quality is an especially critical task since we do not know service exact specifications, developing and testing methodologies. An invocation of a troublesome service can affect all the system. A potential client should be aware of the service behavior before he/she/it starts to exploit it. Among the basic QoS factors are service performance (throughput, response time, latency, transaction time), availability, accessibility, reliability, scalability, exception han-

dling, execution cost, reputation, regulatory, accuracy, integrity, interoperability, security (authentication, authorization, confidentiality, traceability, data encryption, non-repudiation), privacy, network-based factors (network delay, delay variation, packet loss), etc. [12]. Service reputation is an especially interesting property from this list. Kalepu et al. [8] address the problem of subjective perception of reputation due to the lack of performance history. Our proposed solution allows to infer WS rankings from pure statistical data about service invocations based on particular criteria for each client.

An important issue is the way in which a client asks for a service. Currently used by UDDI API keyword-based queries are easy to express but obviously not sufficient. Using WSDL specifications as a query appears more promising. In the case when a client wants to substitute one web service with another without affecting the observable behavior of the system, he/she does not need to write a request. But in general, WSDL specification is redundant and does not contain QoS requirements. Ran [12] suggests SOAP-based requests. In the requirements for WS discovery competitions [2] requests are described in XML format. A similar style can be used to express client's preferences about QoS characteristics and testing samples.

In IR approaches to WS discovery a query consists of keywords, which are matched against the stored descriptions in the UDDI. The Boolean model suffered from either lots or very few returned results. Traditional models like Vector-space model seem to be quite effective. Latent Semantic Indexing (LSI), the prevailing method for small document collections, was applied on the UDDI to capture the semantic associations between services [14]. A suite of algorithms for similarity assessment between two WSDL descriptions were developed [15]. They are based on IR and component matching methods. The WordNet database is applied for semantic analysis of service documentation. According to those experimental results, the methods are neither precise nor robust. Dong et al. [3] describe an approach based on term associations analysis. In [6], WS discovery and composition based on syntactic matching is suggested. Authors propose to build indexes on operations and part names to speed up the matching process. Oh et al. [11] describe the transformation of WS discovery and composition issues into a graph search problem. Web services are compared depending on syntactic matching of input and output parameters whereas data types are ignored.

Retrieval is successful when the information need is satisfied. Therefore, we need to be provided with the user's judgement of the relevance of the found data. Maximillien et al. [9] describe an agent-based system where agents act as proxies to collect information and build a reputation of semantic web services. In [17] trust management mechanism is added to evaluate the credibility of the user reports when predicting service quality. By nature, most of us are not inclined to give feedback. The solution is suggested by a recommendation system that uses implicit feedback from the user by means of analyzing his/her behavior. Birukov et al. [1] present such a system to produce recommendations for web search. In our task, agents can control actions like user search requests, test invocations of web services, bindings, successful service responses or failures, etc.

3 A Framework for WS Discovery

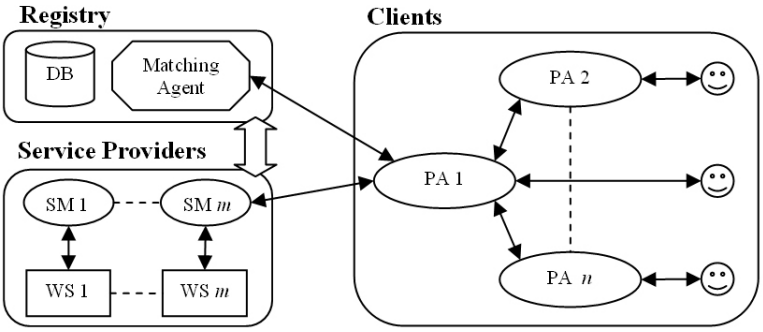

Fig. 1. Web service discovery model. *Personal Agent (PA)* accepts requests from its *client* and sends them to *Registry*. *Matching Agent* provides lists of relevant services and agents that searched for them. *PA* asks agents and *Service Mediators (SMs)* for information, they provide it. *PA* ranks the services and invokes the best one. Both *PA* and *SM* store execution statistics.

In Fig. 1 a basic architecture of our proposed framework is presented. Each client or group of local clients with similar preferences has a dedicated agent which task is to process all activities pertaining to web services (communication with registries, bindings, requests, responses). The *personal agent* accepts a request from its client (human or application), redirects the description of the goal to the *matching agent* which manages the *registry*. The *matching agent* searches for the services that can conceivably satisfy the client's goal. It keeps history of requests, and provides the requester with the information about the agents that searched for the similar services before. Given a list of previous clients, that is the last k agents to which the *matching agent* recommended the service, the *personal agent* can ask them for recommendations. Then two scenarios are possible: a first one, in which the agents that invoked the services provide the requester with their own rankings, and an alternative one, in which they render to it parts of their own history leaving to the requester the task of processing it. The latter approach tries to overcome the problem of specific client's preferences that cannot be provided by external QoS evaluation systems. The *personal agent* can ask first the agents that gave useful statistics before, and only if they do not have necessary information, to apply more extensive interrogation. An open issue is preventing malfeasant agents from affecting the service reputation. If web services use mediators that collect statistics about all their invocations, the *personal agent* may have more reliable data by comparing parameters provided by agents and the *service mediator*. Another advantage is that centralized data is more convenient for evaluation of such QoS factors as availability, reliability, capacity, robustness.

The *matching agent* provides the *personal agent* with the matching score of its request and returned web services. If this score is low, the relevance of

the services is dubious and alternative approaches are needed to check if they correspond with the client's request. Recording of invocation inputs and outputs can help to test the service before usage by new clients. A client sends testing samples to his/her/its agent, and the agent checks if they are among the data it has. If it is so we may strongly believe that the service fits the client's goal and will work properly. But this approach conflicts with security and privacy issues, and may be time and space consuming.

The *personal agent* ranks the web services by combining the scores of matching, recommendations and tests (if they were provided). Before service invocation it asks the client for confirmation. Finally, the best-fit web service is invoked, QoS parameters are measured, and the results are saved. Then, the agent can ask its user if it should always invoke this service for the same query. Such an explicit user feedback can be used to reconstruct the trust policy to other agents or rating algorithm but it is not necessary. If a user keeps invoking a web service, he/she is likely to be satisfied with the service provided. Therefore, repeated usage should have a high weight for QoS evaluation.

4 Conclusions and Future Work

In this paper we have presented a new WS discovery approach. Some subproblems, namely service matching methods, discovery system evaluation and request formats have been discussed. We have proposed a framework for WS selection based on QoS properties collected implicitly by a distributed agent-based system.

We intend to apply the recommendation system presented in [1] for WS discovery. To achieve our goal we need to implement a matching algorithm, and enrich the current functionality of agents with the ability to mediate WS invocations and measure QoS factors. A flexible ranking policy and learning algorithms for agents are also planned as part of our future work.

In [15] pairs of web services are compared by matching their data types, messages and operations. With respect to these approaches we are going to build indexes for provided operations, use semantic matching to eliminate the mismatched operations and apply relaxed structure matching for the filtered subset. We are planning to elaborate the results by trying alternative IR models (LSI, Probabilistic Models, etc.) on WSDL specifications. Another interesting point is that the *matching agent* can learn from experience. It can match each new query across services and across previous requests to them as well.

Our architecture is similar to [9] in a sense that agents control overall communication process between clients and each web service. Several problems are stated in the paper that are also relevant for our task. Among them is the risk tolerance of the user: the invocation of a new service matching the client's goal could be regarded as higher risk than a more mature one. Another issue is that several complementary web services can be found to satisfy the goal. Our current work does not aim at tackling the problem of WS composition but potentially some state-of-the-art approaches can be exploited in order to increase system recall by considering composite services. An application of recommendation sys-

tem for web services spawns a good number of interesting issues, typical for recommendation systems and novel as well. One of them is a new-system cold-start problem, when there are no user ratings of web services (monitored service invocations). In this case, the initial recommendation can be constructed using Google's rating of the provider. For a new-service cold-start problem agents' ratings of the services by the same provider can be used.

References

1. Birukov, A., Blanzieri, E., Giorgini, P.: "Implicit: An agent-based recommendation system for web search", Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems, 2005, pp. 618–624.
2. Blake, M. et al.: "The EEE-05 Challenge: A New Web Service Discovery and Composition Competition", IEEE Computer Society, 2005, pp. 780–781.
3. Dong, X.L. et al.: "Similarity search for web services", Proceedings of VLDB, 2004.
4. Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.: "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?", International Workshop on Web Engineering, 2004.
5. Hausmann, J.H., Heckel, R., Lohmann, M.: "Model-based Discovery of Web Services", Proceedings of the IEEE International Conference on Web Services, 2004.
6. Huang, Sh., Wang, X., Zhou, A.: "Efficient Web Service Composition Based on Syntactical Matching", IEEE Computer Society, 2005, pp. 782–783.
7. Jeckle, M., Zengler, B.: "Active UDDI - an Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation", Web and Database-Related Workshops on Web, Web-Services and Database Systems, LNCS, 2003, pp. 91–99.
8. Kalepu, S., Krishnaswamy, S., Loke, S.-W.: "Reputation = f(User Ranking, Compliance, Verity)", Proceedings of the IEEE International Conference on Web Services, 2004.
9. Maximilien, E.M., Singh, M.P.: "Conceptual Model of Web Service Reputation", SIGMOD Record, Vol. 31, No. 4, Dec. 2002, pp. 36-41.
10. Medjahed, B., Bouguettaya, A.: "A Multilevel Composability Model for Semantic Web Services", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 7, 2005, pp. 954–968.
11. Oh, S.-Ch., Lee, D., Kumara, S.: "Flexible Web Services Discovery and Composition using SATPlan and A* Algorithms", In Modeling Decisions for Artificial Intelligence Conference, 2005.
12. Ran, Sh.: "A Model for Web Services Discovery With QoS", ACM SIGecom Exchanges, Vol. 4, No. 1, 2003, pp. 1–10.
13. Rahm, E., Bernstein, P.: "A Survey of Approaches to Automatic Schema Matching", VLDB Journal, vol. 10, No. 4, 2001, pp. 334–350.
14. Sajjanhar, A., Hou, J., Zhang, Y.: "Algorithm for Web Services Matching", In Proceedings APWeb 2004, LNCS 3007, Springer Verlag, 2004, pp. 665–670.
15. Stroulia, E., Wang, Y.: "Structural and semantic matching for accessing web-service similarity", International Journal of Cooperative Information Systems, Vol. 14, No. 4, 2005, pp. 407–437.
16. The OWL Services Coalition. OWL-S specification version 1.0, November 2003. http://www.daml.org/services/.
17. Vu, Le-Hung. et al.: "Towards P2P-based semantic web service discovery with QoS supports", Workshop on Business Processes and Services, 2005.

Web Service Quality Composition Modelling

Ganna Frankova

DIT - University of Trento, Via Sommarive, 14, 38050 Povo-Trento, Italy
email: ganna.frankova@dit.unitn.it

Abstract. The critical issue of web services success is the ability to compose web services in order to build complex added-value services. In web service compositions both functional and non functional properties, i.e. quality of service, should be taken into account. The quality of service in web service compositions plays a vital role and has opened a wide spectrum of challenges. We survey and analyze various approaches for modelling web service quality composition. Also we overview some ideas of service selection according to the surveyed models. From the analysis it turns out that there are many open issues in the web service quality composition area.

1 Introduction

One of the most thought provoking issues in web services (WS) is that of automatic composing services in order to build complex added-value services. In particular, the quality of service (QoS) compositions plays a vital role and has opened a wide spectrum of challenges.

Nowadays the attention is on quality driven web service discovery [19] and efficient composing [5]. The research is well under way, and most of the focus is on "higher level" issues, i.e., selection and composition. On the one hand, it is important, on the other hand, there is not enough "background", i.e., models for QoS of WS composition are far from ideal ones.

In this paper, we survey and analyze various approaches for modelling web service quality composition. Some ideas of service selection according to the models are presented. From the analysis it turns out that there are many open issues in the quality of web service composition area.

The reminder of the paper is organized as follows. Section 2 discusses related work. Section 3 is devoted to the comparison of the various approaches to WS quality composition modelling with respect to the proposed requirements. Some open issues and concluding remarks are presented in Section 4.

2 Approaches for WS Quality Composition Modelling

Web service QoS issues are gaining attention and have been addressed in a number of recent works. Some approaches are based on the extension of the Web Service Description Language (WSDL) [7] to define not only functional, but also

non-functional properties of the service, e.g., [11]. The problem with this kind of approach is that the QoS definition is tied to the individual operation, rather than to the service as a whole; furthermore, there issue of run-time support is not addressed.

In [20], the authors define QoS for WS by using XML schemas that both service consumers and providers apply to define the agreed QoS parameters. The approach allows for the dynamic selection of WS depending on various QoS requirements. On the negative side, the life-cycle of agreements is not taken into account, and it is not possible to define an expiration for a negotiation.

In [8], the Agreement-Based Open Grid Service Management (OGSI-A) model is presented. Its aim is to integrate Grid technologies with Web Service mechanisms and to dynamically manage negotiable applications and services using WS-Agreement. Recently proposed Web Services Agreement Specification [4] defines the interaction between a service provider and a consumer, and a protocol for creating an agreement using agreement templates. A formal definition of what the semantics of a QoS negotiation is and an extension to make agreements more robust and long-lived are proposed in [2].

The feasibility of using constraint programming to improve the automation of web services procurement is shown in [15]. A predictive QoS model for workflows involving QoS properties is proposed in [6]. In [9], the authors propose a model and architecture to let the consumer rate the qualities of a service. Various approaches and contemporary standards for managing web services can be found in [17]. In addition, the industry has proposed a number of standards to address the issue of QoS: IBM Web Service Level Agreement (WSLA) and HP's Web Service Management Language (WSML) are examples of languages used to describe quality metrics of services [13].

An overview of several of the proposals for web service quality composition modelling follows. For all these approaches we present a realistic car loan example that involves getting a loan and buying a car. We assume that standardized web services interfaces for getting a loan and buying a car already exist. Some ideas of service selection according to the models are also presented.

Continuous-time Markov chains solution. The approach of estimation execution time and cost of a workflow based on continuous-time Markov chains is proposed in [12]. Applying the approach to the running example we take execution cost into account. In order to introduce a simple QoS model, each activity whose start and end is signaled by a service is assigned an execution cost. The cost incurs when the activity is started. Therefore, each transition is labeled with the sum of the costs of all activities which are active in the destination node but not in the source node.

Quality Vector solution. Description of elementary service quality as a quality vector is proposed in [21], a similar solution can be found in [1]. Each component of the vector is a quality criterion for the service. The authors propose to compute quality criteria for composite services by using special aggregation functions.

Applying the approach to the car loan example, a quality vector can be defined as q(s)=($q_{price}(s), q_{duration}(s)$), where $q_{price}(s)$ is price of the service execution; $q_{duration}(s)$ is the expected delay in seconds between the request to the service s sending and the result receiving. The quality vector of a composite service's execution plan p is the following: Q(p)=($Q_{price}(p), Q_{duration}(p)$), where $Q_{price}(p) = \sum_{i=1}^{N} q_{price}(s_i)$; $Q_{duration}(p)$=CPA[1]($q_{duration}(s_1), ..., q_{duration}(s_N)$); N is a number of states in the execution path.

Selection of optimal component services of a composite service can be done with a global planning approach. Furthermore service selection can be formulated as an optimization problem which is solved using linear programming.

Agent-oriented solution. The use of the agent-oriented methodology Tropos to model a wide spectrum of quality of web services properties is proposed in [3]. To model a quality composition of our running example web services, the whole set of interacting services is represented as a multi-agent system.

Ontology-based solution. E.M. Maximilien and M.P. Singh proposed an ontology-based framework for dynamic web services selection [16].

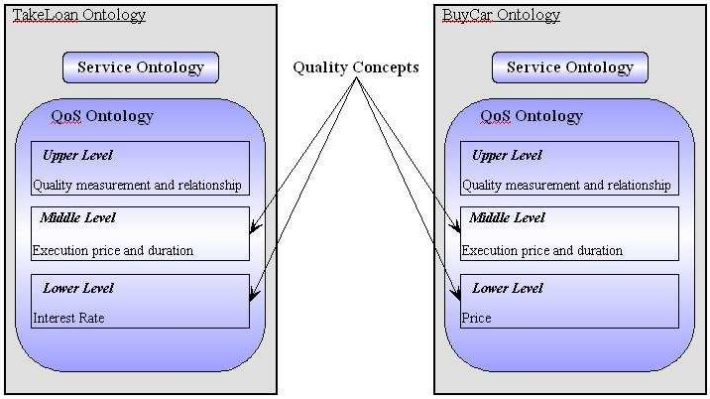

Fig. 1. Ontology-based solution of the loan receiving and a car buying example

The model consists of two-layers: service ontology and QoS ontology. The services ontology relates services to QoS whereas the QoS ontology leads to the quality concepts. The QoS ontology is divided into three ontologies: upper, middle, and lower. QoS upper ontology includes the basic characteristics of all qualities and the main concepts associated with them, e.g., quality measurement and relationships. QoS middle ontology specifies domain-independent quality concepts, e.g., availability, performance, reliability, security. QoS lower ontology is a domain-specific ontology that typically completes the middle one.

Figure 1 shows the application of the approach to the running example. Both for TakeLoan and for BuyCar services service and QoS ontology are presented.

[1] Critical Path Algorithm [18]

The TakeLoan QoS ontology contains the concepts of `Interest Rate`, similarly the BuyCar QoS ontology includes the domain specific concept `Price`. Both domains use concepts of execution price and execution duration depicted in the middle ontology section.

Dynamic service selection calls for an agent-based solution. Using QoS ontology, service agents match advertised quality levels for the consumers with specific QoS preferences.

3 Discussion

Now we can compare the various approaches for WS quality composition modelling with respect to the requirements derived from our exploration of the approaches we surveyed. The table in Figure 2 summarizes the results. The approaches are presented in the columns while the rows show the requirements.

	Markov Chains	Quality Vector	Agent-oriented	Ontology-based
Objective QoS	√	√	√	√
Subjective QoS				√
Run-time support	√		√	
QoS assignment		√	√	
Requirements considering level	Low	Average	High	High

Fig. 2. Comparing approaches for WS quality composition modelling

Objective and Subjective QoS. QoS can be objective, such as reliability, availability, and request-to-response time, or subjective, i.e., focusing on user experience [16]. Objective qualities are taken into account in all approaches, while only ontology-based systems focuses on both types of QoS.

Run-time support. According to the fact that some QoS metrics such as response time can change at run-time and a single value is not appropriate, approaches for WS quality composition modelling should have a run-time support infrastructure. Markov chains- and agents-based approaches provide run-time support.

QoS assignment to composite service. When quality constraints and preferences are assigned to composite services rather than to component services it is easy to reuse the composed service, i.e., use it in another quality driven composition. The table in Figure 2 shows that quality vector-based and agent-oriented solutions allow to assign QoS to composite service.

Quality Requirements. QoS Requirements for Web Services [10, 14] include performance, reliability, scalability, capacity, robustness, exception handling, accuracy, integrity, accessibility, availability, interoperability, security, regulatory, and network-related QoS requirements. Analyzing the approaches one

notes that not all of the requirements are met. Specifically, execution price and time are estimated in [12]. While [21] considers execution price, execution duration, reliability, availability and reputation, and [1] focuses on four quality dimensions, i.e., execution cost, execution time, reliability, and availability. Ontology- and agent-based solutions theoretically deal with on all QoS requirements.

Summarizing, an analysis of the table shows that current QoS models of WS composition are far from ideal, and a number of open challenges exists.

4 Concluding Remarks and Future Directions

In this paper we have presented a survey and analysis of various approaches for modelling web service quality composition. Web service quality composition approaches range from industry standards to more abstract models. An ideal approach should satisfy the identified requirements, but so far there is not an approach fulfilling all requirements.

The survey has highlighted that there is a lot of space for further investigation and innovative research. In particular, web services are widely-used in e-commerce, an area required achieving high level of security. But there is no solution to model security properties of composed services. Further investigation should be devoted to solving security quality integration, as well as run-time support issues. Mathematical modelling approaches, and graph-based solutions are promising fields to look for solutions to this problem, as well as optimization techniques. The author will focus on solving the web service security composition problem by applying optimization techniques.

Acknowledgments

The author thanks Marco Aiello and Fabio Massacci for discussions and assistance in the presented work.

References

1. R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in METEOR-S. In *Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 23–30, Shanghai, China, September 2004.
2. M. Aiello, G. Frankova, and D. Malfatti. Whats in an agreement? An analysis andan extension of WS-Agreement. In *Proceedings of the 3rd International Conference on Service Oriented Computing*, Amsterdam, The Netherlands, December 2005. To appear.
3. M. Aiello and P. Giorgini. Applying the Tropos methodology for analysing web services requirements and reasoning about Qualities of Services. *CEPIS Upgrade - The European journal of the informatics professional*, 5(4):20–26, 2004.
4. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (WS-Agreement). Technical report, Grid Resource allocation Agreement Protocol (GRAAP) WG, 2004.

5. G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. A lightweight approach for QoS-aware service composition. In *Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 36–47, New York City, NY, USA, November 2004.
6. J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3):281–308, 2004.
7. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1, March 2001. `http://www.w3.org/TR/wsdl`.
8. K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based grid service management (OGSI-Agreement). Technical report, Global Grid Forum, GRAAP-WG Author Contribution, 2003.
9. V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. A quality of service management framework based on user expectations. In *Proceedings of the 1st International Conference on Service Oriented Computing*, pages 104–114, Trento, Italy, December 2003.
10. K.-C. Lee et al. QoS for web services: Requirements and possible approaches, November 2003. W3C Working Group Note, `http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/`.
11. D. Gouscos, M. Kalikakis, and P. Georgiadis. An approach to modeling web service QoS and provision price. In *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December 2003.
12. J. Klingemann and K. Wäsch J., Aberer. Deriving service models in cross-organizational workflows. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises*, pages 100–107, Sydney, Australia, March 1999.
13. H. Ludwig. Web services QoS: External SLAs and Internal Policies Or How do we deliver what we promise? In *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December 2003.
14. A. Mani and A. Nagarajan. Understanding quality of service for web services, 2002. `http://www-106.ibm.com/developerworks/library/ws-quality.html`.
15. O. Martín-Díaz, A. Ruiz Cortés, A. Durán, D. Benavides, and M. Toro. Automating the procurement of web services. In *Proceedings of the 1st International Conference on Service Oriented Computing*, pages 91–103, Trento, Italy, December 2003.
16. E.M. Maximilien and M.P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 08(5):84–93, September/October 2004.
17. M. Papazoglou and W.-J. van den Heuvel. Web services management: A survey. *IEEE Internet Computing*, 9(6):58–64, 2005.
18. M. Pinedof. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, second edition, 2001.
19. S. Ran. A model for web services discovery with QoS. *SIGecom Exchanges*, 4(1):1–10, Spring 2003.
20. M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A concept for QoS integration in web services. In *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December 2003.
21. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th International conference on World Wide Web*, pages 411–421, Budapest, Hungary, May 2003.

Extending OWL for QoS-based Web Service Description and Discovery

Kyriakos Kritikos

University of Crete, Heraklion, Greece,
kritikos@csd.uoc.gr

1 Main Theme of Thesis

Web Services (WSs) are modular, self-describing, loosely-coupled, platform and programming language-agnostic software applications that can be advertised, located and used across the Internet. They are viewed as one of the promising technologies that could help business entities to automate their operations on the web on a large scale by automatic discovery and consumption of services. Based on the above reasons, the WS paradigm is being adopted by many companies and individuals and many WSs are being deployed and running.

However, as all of these WSs are advertised in a UDDI-based repository, an unavoidable fact as UDDI is a de-facto standard, the problem of discovering them based on a requester's functional needs becomes crucial. UDDI uses a syntax-based approach for WS description leading to purely syntactic discovery efforts returning imprecise and inaccurate results. **OWL-S** [OWL-S Coalition 2003] and similar joint **Semantic Web** and WS efforts solve the problem of syntactic WS description by using ontologies for describing WSs. Ontologies provide meaning to concepts and relationships between them, leading to semantic **WS Discovery** algorithms, which provide more precise and accurate results.

But even if all the advertised WSs satisfying a requester's functional needs are returned, many results may be produced. So a non-functional concept is needed that will differentiate between the functionally equivalent WS advertisements. This concept is **quality of service** (**QoS**). QoS is closely related with the performance of a WS as well as with other features of a WS that bear on its ability to satisfy stated or implied needs. Therefore it has a substantial impact on users' expectations from a service. Thus WS descriptions must be enhanced with QoS descriptions. Additionally, WS discovery algorithms should perform QoS-based matchmaking and selection in order to produce fewer ranked results.

Unfortunately, all the current research efforts fail in correctly describing QoS for WSs. **Semantics** seems to be missing from the QoS description of a WS leading to purely syntactic QoS-based WS matchmaking and selection algorithms. But even if semantics is introduced, QoS description is not rich enough and not quite extensible. So the **main issue** of this PhD thesis is the **rich, extensible, and semantic description of QoS for WSs**. Additionally, **new** QoS-based WS **matchmaking and selection algorithms** must be **devised** or the **current best** should be **extended** in order to take advantage of this enhanced semantic QoS description.

2 Motivation for Research

After reviewing several definitions for QoS for WSs, we consider **QoS** for a WS as "*a set of non-functional characteristics/attributes that may impact the quality of the service offered by the WS*". If a WS is advertised to have certain values (or range of values) in these QoS attributes, then it is said that this WS **conforms** to a certain **QoS level**. In this section we explain the reasons for incorporating QoS in WS description.

According to [Cardoso et. al. 2004], several researchers have identified Web Processes (WPs) as the computing model that enables a standard method of building Web-services applications and processes to connect and exchange information over the Web. For organizations, the ability to characterize WPs based on QoS has **four distinct advantages**. **First**, it allows organizations to translate their vision into their business processes more efficiently, since WPs can be designed according to QoS metrics. For e-commerce processes it is important to know the QoS an application will exhibit before making the service available to customers. **Second**, it allows for the selection and execution of WPs based on their QoS, to better fulfil customer expectations and requirements. **Third**, it makes possible the monitoring of WPs based on QoS to assure compliance both with initial QoS requirements and targeted objectives. QoS monitoring allows adaptation strategies to be triggered when undesired metrics are identified or when threshold values are reached. **Fourth**, it allows for the evaluation of alternative strategies when adaptation becomes necessary. **It is essential** that the services rendered follow customer specifications to meet their expectations and ensure satisfaction. *Customer expectations and satisfaction can be translated into the quality of service rendered.* Organizations have realized that quality of service management is an important factor in their operations.

As WPs are composed or single WSs, all the above advantages of QoS management of WPs also apply to WSs. So WSs can be designed and implemented according to QoS metrics (properties). They can also be discovered and selected based on their QoS capabilities. In addition, they can be monitored in order to reassure the promised QoS levels to the customers. Moreover, monitoring of QoS for WSs can trigger adaptation strategies when undesired metrics are identified, threshold values are reached, network or software or hardware errors happen. Now, we will closely examine the advantages of QoS description (management) in other non-basic activities/functions of the Service Oriented Architecture.

After the process of **WS Selection**, the requester chooses the best WS from an ordered WS advertisement list. However, even if WS clients find the appropriate WS, they are not confident that the WS's described QoS levels will actually be delivered during **WS execution**. For this reason, the WS client and provider **enter a multi-step negotiation phase**, where they try to agree on a trusted third-party entity monitoring QoS levels delivered, on the penalties that will be imposed when one of the two main parties does not keep up with its promises, and on the validity period of the promises. The **result** of this negotiation phase is a **contract** or a **Service Level Agreement** (**SLA**) document that will give confidence and trust to the entities providing and consuming the service and will

lead and guide the process of WS Execution. If agreement is not met, the negotiation is stopped and the WS client contacts the next WS from the returned list of the WS Selection phase.

When **composing** a WS, **component services** are associated to the individual tasks of the composite WS and are invoked during each execution of the WS. However, the number of services providing a given functionality may be large and constantly changing and some of these services will not always be available due to network problems, software evolution and repair, and hardware problems. One solution to this problem is given at **design time** by QoS-based WS discovery. Another solution is the **runtime selection** of component services, during the execution of a composite WS, based on quality criteria (i.e. constraints and preferences) and following a local [Benatallah et. al. 2002] or a global [Zeng et. al. 2003] selection strategy. In the latter case, **service selection** is formulated as an **optimization problem** and linear programming is used to compute optimal execution plans for composite services.

3 Review of Related Work

In this section, the current research approaches for QoS-based WS Description and Discovery are described and their deficiencies are analyzed.

The **Web Service Description Language** (**WSDL**) and **UDDI** WS *standards* are **syntactical** approaches that do not express the QoS aspect/part of WS Description. While **OWL-S** is a *standard* **semantic** approach for WS Description, it does not describe QoS offers or demands as it only contains an attribute used for rating a WS.

[**Tosic et. al. 2002**] argue that for the specification of constraints for QoS metrics/attributes, five ontologies must be developed from which the most important (the top one) is the metrics ontology. They describe the structure and involved elements in four out of the five ontologies but they did not develop any ontology. In addition, the requirements specified are incomplete as each from the four aspects of QoS description needs further analysis.

In [**Shuping Ran 2003**], an extension to UDDI is proposed that represents description of QoS information about a particular WS. However, there is no actual description of the contents of this extension apart from its structure. Moreover, it relies on the UDDI (model), so it can be used only for syntactic matchmaking of offers and demands.

In [**Maximilien and Singh 2002**], an architecture and a conceptual model of WS reputation (QoS) (which encloses a QoS attributes model) are presented. However, the reputation of a WS is calculated and not it's QoS. In addition, not only concepts like QoS constraints and QoS offers and demands are not modeled but also the QoS metrics model is not rich enough.

Work described in [**Tosic et. al. 2003**], which presents the **Web Service Offerings Language** (**WSOL**), proposes that a WS must offer different classes of service in order to satisfy a greater amount and type of customers and in order to deal successfully with situations where there is a variation in QoS due

to network problems or mobility reasons. This work comes with the following shortcomings: (a) no separation and integration of constraint dimensions; (b) no specification of a QoS demand; (c) the metrics ontologies are not yet developed.

The research effort described in [**Tian et. al. 2003**] analyzes what must be enclosed into the QoS information for a WS request or advertisement with the help of a QoS ontology. However, not only there is not a complete and accurate description of QoS constraints but also metrics ontologies are not developed but just referenced.

In [**Zhou et. al. 2004**], DAML-S WS description language is extended to include a QoS specification ontology. In addition, a novel QoS matchmaking algorithm is proposed, which is based on the concept of **QoS profile compatibility**. The deficiencies of this research effort are the following: (a) The metrics model is not rich enough; (b) QoS metrics have the set $\mathbb{N}^+$ as their range; (c) DL reasoners are slow and do not support the most complex mathematical expressions.

The research effort described in [**Martín-Díaz et. al. 2003**] uses a symmetric QoS model expressing mathematical constraints for QoS metrics. However, **semantics is missing** leading to syntactic matchmaking and selection algorithms. Before matchmaking, a QoS specification is transformed to a **Constraint Satisfaction Problem** (**CSP**) [Hentenryck and Saraswat 1996] which is checked for **consistency** (if it has any solution). Matchmaking is performed according to the concept of **conformance** (if every solution of offer is a solution of demand). Concerning WS Selection, the (QoS) score of a WS advertisement is expressed as a **Constraint Satisfaction Optimization Problem** (**CSOP**) [Hentenryck and Saraswat 1996] where from all solutions to the CSP of an offer we try to find the one that minimizes the weighted sum of the weight of each metric multiplied with its utility assessment value. Unfortunately, CS(O)Ps can have non-polynomial solutions when there are non-linear expression at QoS constraints.

4 Future Work of the Thesis

Based on the previously analyzed research work, we propose the following steps that must be taken to address the issues of the thesis; the fulfilment of which will lead to correct, efficient and accurate QoS-based WS Description and Discovery processes. Some of them have already been performed while the other will be dealt with in the future. These steps are:

Requirements for QoS-based WS Description: This step has already been taken. We have come up with the following requirements:

- Extensible and formal semantic QoS model
- Standards compliance
- Syntactical separation of QoS-based and functional parts of service specification
- Both requester and provider QoS specification
- Refinement of QoS specifications (extensibility, reusability)

- Fine-grained QoS specification (QoS specs for the whole WS and its parts)
- Extensible and formal QoS attribute/metric model which must at least specify: **(a)** The value set of the attribute; **(b)** The domain of knowledge of the attribute; **(c)** The relationship of the attribute with other attributes; **(d)** The association of the attribute with a unit, a measured property and a measurement function; **(e)** A functional description of how a QoS attribute of a complex WS can be derived from the corresponding QoS attribute of the individual WSs that constitute the complex one.
- Classes of service (an advertisement should present many offers)

QoS-based WS Description : Based on the above requirements, a QoS-based WS ontology has been developed with the name **OWL-Q**. This ontology extends **OWL-S** (standard) WS Description ontology and is carefully separated into many facets, each capturing one aspect of QoS WS (metric) description except from the basic one that associates OWL-S elements with OWL-Q elements. This ontology satisfies the above requirements and enables the semantic matchmaking of QoS advertisements and offers.

Semantic QoS metric matchmaking: Based on the OWL-Q ontology, we have developed an algorithm that semantically matches QoS metrics of demands and offers. This algorithm can be used as a building block for the QoS-based WS matchmaking and selection algorithms. For matchmaking simple metrics, this algorithm compares the type and measurement directive of the metrics, while for composite metrics it also compares the metrics' measurement functions.

Develop a new or extend an old QoS-based WS matchmaking algorithm: We have extended the matchmaking algorithm proposed in [Martín-Díaz et. al. 2003] in order to incorporate the semantic QoS metric matchmaking algorithm. To be more specific, when transforming QoS offers and demand to CSPs, we are careful to assign same metrics to same CSP variables and to perform unit transformation of the same metrics.

Develop a new or extend an old QoS-based WS selection algorithm: We have extended the selection algorithm proposed in [Martín-Díaz et. al. 2003] in order to incorporate the semantic QoS metric matchmaking algorithm. To be more specific, when transforming QoS offers to CSOPs, we are careful to assign same metrics to same CSP variables and to perform unit transformation of the same metrics. We have also changed the scoring function: now this function is the partial sum of the minimum and maximum assessment sums. That is we solve two CSOPs for the same offer and then we perform a partial sum of their results.

Implement these matchmaking and selection algorithms: This is a step under development. The implementation uses an OWL inference engine for the semantic QoS metric matching and the ECLiPSe [Novello and Schimpf 1999] engine for solving CS(O)Ps.

Formal evaluation of the above algorithms: The above three algorithms should be formally evaluated in order to prove that they are efficient, quick, accurate and precise.

Extend the ontology and the algorithms: After performing all the above steps, OWL-Q ontology should be extended to include other non-functional descriptions of WSs (mainly contextual ones) and its design must be finalized. In addition, the QoS-based matchmaking algorithm must be extended in order to distinguish between soft and hard non-functional constraints.

Tools: GUIs and other utilities should be constructed that will help the user in describing and discovering WSs.

References

BENATALLAH, B., DUMAS, M., SHENG, Q. Z., AND NGU, A. 2002. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *Proc. of ICDE'02*, IEEE Computer Society, pages 297–308, San Jose.

CARDOSO, J., SHETH, A., MILLER, J., ARNOLD, J., AND KOCHUT, K. 2004. Quality of Service for Workflows and Web Service Processes. Journal of Web Semantic (to appear in 2004).

HENTENRYCK, P., AND SARASWAT, V. 1996. Strategic directions in constraint programming. *ACM Computing Surveys*, 28(4), December 1996.

MARTÍN-DÍAZ, O., RUIZ-CORTÉS, A., BENAVIDES, D., DURAN, A., AND TORO, M. 2003. Automating the procurement of Web Services. In *Proc. of the International Conference on Service Oriented Computing (ICSOC'03), Springer LNCS 2910*, pages 91–103, Berlin, Germany.

MAXIMILIEN, E.M., AND SINGH, M. 2002. Conceptual model of web service reputation. SIGMOD Record, ACM Special Interest Group on Management of Data (2002) 36–41.

NOVELLO, S., AND SCHIMPF, J. 1999. ECLiPSe Embedding and Interfacing Manual.

OWL-S COALITION (2003): OWL-S 1.0 Release. `http://www.daml.org/services/owl-s/1.0/`.

RAN, SHUPING. 2003. A Model for Web Services Discovery with QoS. A*CM SIGecom Exchanges*, 4(1):1–10.

TIAN, M., GRAMM, A., NABULSI, M., RITTER, H., SCHILLER, J., AND T. VOIGT, T. 2003. QoS Integration in Web Services. Gesellschaft fur Informatik DWS 2003, Doktorandenworkshop Technologien und Anwendungen von XML (Ph.D. students workshop Technologies and Applications of XML), Berlin, Germany, Oct. 2003.

TOSIC, V., ESFANDIARI, B., PAGUREK, B., PATEL, K. 2002. On Requirements for Ontologies in Management of Web Services. In *Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web - WES* (at CAiSE'02), Bussler, C. et al. (eds.), Toronto, Canada, May 2002. Springer-Verlag, Lecture Notes in Computer Science (LNCS), No. 2512, pp.237–247.

TOSIC, V., PAGUREK, B., AND PATEL, K. 2003. WSOL — A Language for the Formal Specification of Classes of Service for Web Services. In the *Int. Conf. on Web Services (ICWS03)*. Las Vegas, USA, June 23–26, 2003, CSREA Press, pp. 375–381.

ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., SHENG, Q. 2003. Quality driven web services composition. In: *WWW 2003*: 411–421.

ZHOU, C., CHIA, L-T., AND LEE, B-S. 2004. DAML-QoS Ontology for Web Services. In: *ICWS 2004*: 472–479.