Modeling QoS characteristics in WSMO

Ioan Toma
Digital Enterprise Research Institute (DERI Innsbruck)
Technikerstrasse 21a, 6020 Innsbruck, Austria
ioan.toma@deri.org

Douglas Foxvog
Digital Enterprise Research Institute (DERI Galway)
National University of Ireland, Galway, Ireland
doug.foxvog@deri.org

Michael C. Jaeger
TU Berlin, FG FLP, FR6-10,
Franklinstrasse 28/29,
D-10587 Berlin, Germany
mcj@cs.tu-berlin.de

ABSTRACT

Service oriented architectures (SOAs) are becoming widespread solutions for realizing distributed applications. They promote a service view of the world in which functionalities exposed as services by different companies are assembled and reused in a standardized manner. Services are the core building blocks of SOAs and therefore modeling various aspects of services becomes a fundamental challenge. Among these aspects, quality-of-service (QoS) need to be addressed given the high dynamism of any SOA-based system. This paper introduces the basic steps of modeling QoS characteristics of services with the Web Service Modeling Ontology (WSMO) in order to provide a QoS-aware SOA. It discusses the current limitations of modeling QoS characteristics with WSMO and proposes a set of approaches towards a richer QoS modeling support. Each approach is analyzed in terms of complexity and the advantages and disadvantages of each approach are discussed.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: Miscellaneous

General Terms

Design

Keywords

QoS, Non-Functional Properties, Modeling

1. INTRODUCTION

Electronic services and the service-oriented architecture (SOA) are emerging paradigms for the IT infrastructure of today's enterprises. An SOA meets special demands of businesses to realise their processes, because services match the process orientation of modern businesses. Basic cornerstones of today's services are standardised interface descriptions and standardised invocation protocols. The standardisation enables the interoperation between heterogeneous computer systems. Heterogeneity is the reality in the IT-landscape of today's business. The standardisation of interface and invocation descriptions allows the facilitation of an implementation-independent infrastructure which is an economic advantage for businesses. Thus, there is strong business interest for an SOA to establish a common platform that integrates the different systems (cf. Huhns and Singh [15]).

Along with the emerging use of SOAs, the need rises to consider quality-of-service (QoS) when running business processes. The volatility of the offered service quality is an inherent characteristic of distributed systems. Services may appear or disappear, or the underlying infrastructure may show volatile behaviour regarding the connection quality. Thus for businesses, so-called service level agreements (SLA) become important in order to preserve business interests. The basic concern of an SLA is to negotiate a guaranteed level of provided QoS. Such QoS could cover the cost of service execution as well as the response time of a service invocation or the availability of services.

This paper introduces the basic steps of modeling QoS characteristics of services with the Web Service Modeling Ontology (WSMO, [18]) in order to provide a QoS-aware SOA. WSMO, along with its associated language, the Web Service Modeling Language (WSML, [4]), provides the means to describe different characteristics of SOA services in a machine understandable way. It focusses on two aspects:

- **Automated service mediation.** Computers systems should identify suitable services automatically. The approach proposed by WSMO is to apply semantic descriptions and formal modeling of services to let software systems automatically decide whether a service is suitable to take part in a business process or not. The concept of mediation includes the transformation of functionality to translate between the provider's offerings and the requirements of a service requester.
- **Service execution infrastructure.** Another main aspect of WSMO is to provide an infrastructure that allows not only the description and mediation of services but also facilitates the provision of services. This infrastructure is called Web Service Modeling Execution Environment (WSMX, [9]).

As mentioned above, business interests require facilities to guarantee a constant level of QoS. Our aim is to extend the WSMO approach with capabilities to support and process QoS characteristics. Processing the QoS has two main purposes:

- **Matching Requirements.** In the mediation phase, a QoS-aware mediation environment can capture the requirements of the service requester more precisely. These would be business requirements requirements if the services were part of a business process.

MW4SOC '06, November 27-December 1, 2006 Melbourne, Australia

- **Ensuring Continuous Level of QoS.** Based on the agreed level of QoS or SLA a service execution infrastructure monitors QoS violations and can establish recovery activities if required. Such activities keep the level of guaranteed QoS, even if individual services fail. If services fail, they should be dynamically substituted with the appropriate replacements.

To cover this overall goal, this paper presents a set of possible approaches to express QoS characteristics of services in WSMO and WSML. Although different approaches exist for modeling and expressing QoS, none of these proposals are tailored for the use in WSMO. The related efforts will be discussed further in Section 5.

The remainder of the paper is organised as follows: Section 2 will explain basic patterns of processing. Section 3 will introduce the basic concepts of WSMO and WSML, with a focus on current support for modeling QoS characteristics. Section 4 will discuss possible extensions and integration approach for QoS support in WSMO and WSML. After the related work in Section 5, our conclusions will be presented in Section 6.

2. QOS PROCESSING IN AN SOA

The current state-of-the-art proposes three main ways to process the QoS in an SOA for the purpose of service mediation: *a)* using a combined broker, *b)* using a dedicated broker or *c)* direct negotiation between service importer and exporter. All these patterns have in common that a service provider published its QoS offerings while the submission of QoS requirements by the service requester can be regarded as optional. In detail, the three main patterns are as follows:

- **Combined broker.** By this pattern, a broker or a service discovery is extended to process QoS information for trading services. OASIS has proposed a service discovery specification which can be used for this purpose named UDDI (Universal Description Discovery and Integration, [22]). An extended UDDI (cf. [1]) processes the QoS parameters provided by a service provider. When a service requester queries the broker, QoS requirements can be processed for the identification of services.

- **Separate QoS-broker.** In the this pattern, a dedicated broker performs the QoS-based selection of the trading process (cf. [13, 23]). A motivation for establishing a dedicated broker infrastructure exists, if monitoring during run-time is required. A service repository concentrates on the discovery part of the trading process while a QoS-broker monitors the provided QoS in order to apply dynamic adaptations.

- **Direct negotiation.** In this pattern, the QoS information is processed individually and decentrally. A service requester agrees with the provider on a service level agreement. A requirement from the service requester can be considered optional. Such proposals require a negotiation protocol and a QoS specification language; so the requester negotiates an SLA with the provider. Both parties define in this SLA the QoS that the service exporter must provide. Examples for SLA languages are the Web Service Level Agreement Language by IBM (WSLA, [12]) and the Web Service Offerings Language by Tosic et al. (WSOL, [21]).

These three patterns outline the basic arrangements in a QoS-aware SOA. In the WSMX architecture, a dedicated software component is specified that performs the QoS negotiation part. The WSMX architecture provides a service discovery component for the functional matchmaking of service descriptions. In addition, it provides the component named *service selection component* (cf. [5, Section 2]), which selects matched services by their non-functional preferences which includes the QoS. The service selection component will be the component that processes the QoS description as proposed in this work. Additionally, a ranking mechanism for Web services that uses QoS descriptions can be implemented as part of this component.

Besides the different patterns for processing QoS, QoS characteristics for the use with services are another important issue. Menasce mentions response time, throughput, security and availability as relevant characteristics [14]. Zeng et al. present a framework for the QoS-aware composition of Web services [25]; their discussion covers the QoS characteristics price, duration, reputation, success rate, and availability. Patel et al. discuss the modeling of Web services and the creation of service descriptions which involved different QoS characteristics [17]. Their selection is divided into two main categories: The first category consists of the latency(response time), throughput, reliability, and cost. The other category is named internet-specific and consists of availability, security, accessibility. The contributions from Ludwig et al. (WSLA, [12]) and Tosic et al. (WSOL, [21]) do not mention particular QoS categories or characteristics.

Obviously, a basic set of QoS characteristics can be considered relevant for the use in an SOA. The design of languages for QoS description can focus on these characteristics, should not be limited to them. The QoS characteristics considered in WSMO are discussed in Section 3.3. The set of QoS is extensible and the current extensions proposed in this paper do not restrict the QoS characteristics that can be described.

3. THE WSMO APPROACH TO SEMANTIC WEB SERVICES

In this section we give a short overview of the Web Service Modeling Ontology, the conceptual model for describing Semantic Web services, and of the Web Service Modeling Language, the language for describing services based on WSMO model.

3.1 WSMO

The **Web Service Modeling Ontology** is one of the major initiatives in Semantic Web services area. WSMO provides an overall framework for Semantic Web services that aims at supporting automated Web service discovery, selection, composition, mediation, execution, monitoring, etc. WSMO inherits a set of design principles from the Web Service Modeling Framework (WSMF, [6]). Among these principles, two have a major influence: *(1) Principle of maximal de-coupling*: all WSMO components are specified autonomously, independent of connection or interoperability with other components and *(2) Principle of strong mediation*: the connection and interplay between different components is realized by Mediators that resolve possible occurring heterogeneities between the connected components. Additionally every WSMO component description may include an extensible set of non-functional properties, based on the Dublin Core Metadata Set [8]. WSMO defines four top-level notions related to Semantic Web services:

- **Ontologies**: are formal explicit specifications of shared conceptualizations. They define a common agreed upon terminology by providing concepts and relationships among the set of concepts from a real world domain. Such terminologies are then used within all other WSMO elements.

- **Goals**: are descriptions of the objectives a client may have

when consulting a service in terms of functionality, behavior and quality of service.

- **Web services**: are descriptions of services that are requested by service requesters, provided by service providers, and agreed between service providers and requesters.

- **Mediators**: address the heterogeneity problem that occurs between descriptions at different levels: *data level* - different terminologies, *protocol level* - different communication behavior between services, and *process level* - different business processes. WSMO defines four types of mediators: *OO Mediators* connect and mediate heterogeneous ontologies, *GG Mediators* connect Goals, *WG Mediators* link Web services to Goals, and *WW Mediators* connect Web services resolving mismatches between them.

3.2 WSML

The **Web Service Modeling Language** is a formal language for describing ontologies, goals, Web services and mediators. WSML follows the WSMO conceptual model being based on a set of well-known logical formalisms including: Description Logics [2], Logic Programming [11], F-Logic [10] and First Order Logic. These formalisms are taken as starting points for the development of a number of WSML language variants. WSML has a set of five variants: WSML-Core, WSML-Flight, WSML-Rule, WSML-DL and WSML-Full. WSML-Core is based on the intersection of Description Logics and Logic Programming, more precisely on Datalog programs. It has the least expressive power but provides a low formal complexity and is decidable. By extending WSML-Core in the direction of Logic Programming with default negation, cardinality constraints, n-ary relations with arbitrary parameters and meta-modeling features a new language, WSML-Flight, is defined. A further extension in the same direction with function symbols results in a new language variation called WSML-Rule. WSML-Rule no longer requires safety of rules. The only differences between WSML-Rule and WSML-Flight are in the logical expression syntax [4]. Extensions of WSML-Core extension to a full-fledged description logic resulted in WSML-DL. WSML-Full is based on First Order Logic and acts as umbrella language, unifying all the above varieties.

3.3 Current support for QoS modeling in WSMO/WSML

Describing QoS characteristics of a service in WSMO, currently relies on, Dublin Core Metadata Initiative [24], which provides a wide range of non-functional properties. Such properties can be attached to a service description or any other WSMO element. WSMO recommends a set of non-functional properties for each WSMO element of a service description. For example the recommended non-functional properties for services are: *accuracy, contributor, coverage, creator, date, description, financial, format, identifier, language, network-related QoS, owner, performance, publisher, relation, reliability, rights, robustness, scalability, security, source, subject, title, transactional, trust, type,* and *version*.

A closer look at non-functional properties proposed by WSMO revels that there are two categories of non-functional properties. On one hand properties such as *reliability*, *scalability* and *security*, mentioned in the previous section as typical QoS characteristics in a SOA, relate strictly to a service and capture constraints over its functional and behavior aspects [3]. QoS properties are part of this category. On the other hand properties such as *contributor*, *creator* and *date* are rather used to add extra information about the service description itself and do not provid constraints over what a service can do, nor how it can do it. The entire set of properties belonging to this second category includes *contributor, creator, date, identifier, owner, publisher, subject, title,* and *version*.

Both types of non-functional properties are important for service and services descriptions and therefor they are both considered in WSMO. However, the current support in WSMO to model, attach and reason with QoS descriptions of a service is rather limited. WSMO does not provide a model for non-functional properties in general not for QoS in particular. Using WSML [4], the Web Service Modeling Language, one can only assign simple values to the non-functional properties of a WSMO elements. Such a value can be any identifier in WSML and thus it can be an IRI, a data value, an anonymous identifier or a list of any of these. To overcome these limitations we propose in Section 4 a set of solutions that will allow non-functional properties in general and QoS characteristics in particular to be better modelled.

4. EXTENDING WSMO/WSML WITH QOS SUPPORT

Extending the Web Service Modeling Ontology (WSMO) and its associated language Web Service Modeling Language (WSML) with QoS support is the focus of our work. This overall goal generates two important challenges: (1) how to model QoS and (2) how to attach QoS characteristics to services and goals.

For the first challenge one possible general solution is to **define a set of QoS ontologies** that are used afterwards when the QoS characteristics of services are specified. This approach does not require any extensions of the WSML language. It is based only on the usage of ontologies that provide models for QoS. Furthermore, the ontology can be imported and concepts referring to a specific QoS can be instantiated and used in the service description. We have already started to define a set of QoS ontologies [20] in WSML based on the models provided in [16]. These ontologies provide formal conceptualization for Web service QoS like availability, security, etc.

For the second challenge one possible general solution is to **treat QoS, and non-functional properties in general, as normal attributes** for services and goals. In this case QoS properties become part of the logical model of WSML and thus reasoning over these descriptions is possible. More precisely, if services or goals can be seen as instances of concepts then following this extension associated QoS characteristics will be modeled as attributes.

A set of concrete solutions for integrating QoS characteristics in particular and non-functional properties in general in WSMO and writing them in WSML are proposed below. These solutions are concrete implementations that address the second challenge mentioned above. The term non-functional property of a service is being used in this section as a broader term than QoS, *QoS properties* are a subclass of the general class *non-functional properties* defined by WSMO.

- One method for defining non-functional properties for services, goals, mediators and ontologies would be to specify each property through the use of a relation. Relations in WSML need not restrict their arguments to instances; they can be defined with an arity and no restriction on their arguments. The relation, *nfpOf* could be defined as follows to relate the value of properties to such WSMO components:

```
relation nfpOf/3
    nfp
    dc#description hasValue
        "nfpOf(WSMO_Component, Property_1,
            this\_Value) means that
```

```
        the value of the Property_1
            property of the specified WSMO
        component is thisValue."
    endnfp
```

Using such a relation, one could state non-functional properties without constraint:

```
nfpOf(Goal01, serviceCharge,
        _#[hasAmount hasValue 0.02,
        hasCurrency hasValue cur#Euro]
        memberOf price#AbsolutePrice)
nfpOf(OOMediator17, usedMappingLanguage,
    lang#TRIPLE)
nfpOf(Choreography34, usesMediator,
    OOMediator17)
```

This approach has the following advantages:

- No need for adding vocabulary to WSML
- Any property may be used as a non-functional property

This approach has one major disadvantage. There are no restrictions on the value of arguments to *nfpOf*. Thus invalid *nfpOf* relations cannot be detected in this manner. More precisely:

- The second argument is not restricted to being an attribute.
- Values may be presented for the third argument which violate argument restrictions on property specified in second argument.

However, rules could be written to enforce some of these restrictions.

- A second approach for defining non-functional properties for services, goals, mediators, ontologies, or any other WSMO element would be to define a concept *NfpSet* with an attribute *isAbout* that once instanced will point to a WSMO element instance. Non-functional properties are attached to a WSMO element by refining the *NfpSet* concept. Axioms can be used to restrict instances of non-functional properties. An example of how this approach can be implemented is provided below:

```
concept NfpSet
    isAbout ofType _iri

concept WebServicePriceNFPs subConceptOf
     NfpSet
  nfp
    dc#relation hasValue EuroWebServiceOnly
  endnfp
  hasPrice ofType price#Price

axiom EuroWebServiceOnly
    definedBy
        ?x memberOf WebServicePriceNFPs
             equivalent
        ?x[hasPrice hasValue _#[hasCurrency
             hasValue cur#Euro]] memberOf
            price#Price.

instance MyServiceNFPs memberOf
     WebServicePriceNFPs
   isAbout hasValue MyService   // a Web
        Service
   hasPrice hasVslue _#[hasAmount hasValue
        0.02, hasCurrency hasValue cur#Euro
        ]
   memberOf price#AbsolutePrice
```

Constraints over the non-functional properties instances can be defined using axioms. Ontologists are advised to include the relation between the concept and the axioms related to the concept in the non-functional properties through the property *dc#relation*.

This approach has the following advantages:

- No need for adding vocabulary to WSML.
- Logical expressions can be define for non-functional properties and can be attached to a WSMO element (service, goal, etc.). This is done by using the non-functional property *dc#relation*.
- The set of non-functional properties is not an explicit, finite set. An open set of non-functional properties can be attached to a WSMO service or goal as attributes of *NfpSet* sub-concepts.

This approach has the following disadvantages:

- Attaching non-functional properties to WSMO elements by defining concepts as *WebServicePriceNFPs* and *NfpSet* seams a bit artificial. A better way to model is by using a relation.
- The *dc#relation* which appears to connect the definitional axiom to the instance of *NfpSet* is actually a mere comment. This can lead to version control problems if the axiom is renamed or new definitional axioms are added without new *dc#relation* statements being added.

- The third approach for integrating non-functional properties in WSMO/WSML is to model non-functional descriptions of services or goals in a way similar to which capabilities are currently modelled in WSMO/WSML. A service is an entity which provides a functionality (e.g. given a date, a start location, a destination and information about a client a service can book a ticket for the desired trip), but in the same time a service can be seen as an entity which provides one or more non-functional properties (e.g. given a particular type of client a service charges a particular price, etc.). Non-functional properties are defined using logical expressions same as pre/post-conditions, assumptions and effects are being defined in a capability. A simplified model of a WSMO service following this approach is:

```
webService
    capability idCapability
        precondition definedBy axiom1
        postcondition definedBy axiom2
        assumption definedBy axiom3
        effect definedBy axiom4
    nonFunctionalProperty idNFP
        definition definedBy axiom5
```

This approach has the following advantages:

- The set of non-functional properties is not an explicit, finite set. Users of WSMO/WSML can define and attached an open set of non-functional properties to a goal or a service.
- Non-functional properties models are attached to services and goals in the same way as capabilities are.

This approach has the following disadvantages:

- A major disadvantage of this approach is that the WSML syntax has to be extended.

5. RELATED WORK

Another approach for the semantic description of Web service is the OWL Services ontology(OWL-S, [19]) which uses the Web Ontology Language (OWL). The authors of this language propose expressing QoS parameters or constraints, depending on whether a service request or an offer is described, in two main ways: First, a concept that covers the notion of resource consumption is proposed. This concept allows the description of QoS characteristics such as cost and response time, but hardly matches availability or security. To cope with QoS characteristics that do not match the notion of resources, a property of the so-called service profile is provided. In this way, attributes of a service can be defined that express QoS statements. Because the use of quality rating statements is not specified, machine-based reasoning can only be provided with proprietary conventions.

Besides the area of semantic web services, a couple of approaches cover QoS description for other reasons: The WSLA by Ludwig et al. [12] has already been mentioned; its purpose is to provide descriptions for facilitating an SLA negotiation. The language provides the basic concept of a service, which relates to the concept of a Web service in WSMO. WLSA also offers concepts to model the role of different parties involved within a particular SLA. WSMO provides more concrete definition with the concepts of goals, Web services and mediators in the setup of an SOA. As its main goal, the language allows the definition of (QoS) guarantees. WSMO has a broader focus, it also includes basic concepts of service descriptions and description about involved parties. Concepts of WSMO allow the description of general non-functional and behavior characteristics.

The same consideration applies to QML by Frohlund and Koistinen [7] when compared with WSMO and its extensions. QML offers a comprehensive set of description elements which allow the description of QoS to establish SLAs, which are called "contracts" in their work. In contrast to WSLA, the QML provides a concrete set of QoS characteristics and elements for more detailed definitions of QoS parameters, e.g. the percentile statement. Contrary to WSLA, which directly proposes an XML notation, Frolund and Koistinen provide an abstract syntax which would allow different notations besides XML. The QML proposal includes an extension for UML which indicates that QML has its primarily focus on modeling QoS, and can be used in various ways: to express QoS in software models and to provide the foundation for the negotiation about QoS in order to form contracts.

The proposed WSMO extensions of this paper are influenced by WSLA and QML. However, they were not fully considered as basis for WSMO extensions. The language syntax and the interpretation of the main concepts of those proposals are different than in WSMO and WSML. Thus, this paper proposes extensions specifically tailored for WSMO.

6. CONCLUSIONS AND FUTURE WORK

This paper has introduced the basic steps of modeling QoS characteristics of services with the Web Service Modeling Ontology. An analysis of current support for this task has shown that WSMO currently provides limited support for describing QoS characteristics of services. Furthermore the constructs used (key-value pair attributes) are pure syntactic constructs without any formal semantics captured in the framework. Motivated by this analysis, a set of approaches for modeling QoS characteristics of services in WSMO, were proposed. The first two approaches advance the QoS modeling support, without any changes to the conceptual model (WSMO) or to the language (WSML). However the supported expressivity, in this case, is rather limited and expressing QoS characteristics in this way seams quite artificial. The third approach, promotes a more natural way to deal with QoS characteristics of services by considering QoS characteristics of services at the same level as service capabilities. However this approach requires more extensions to WSMO/WSML.

As the modeling of QoS characteristics of services in WSMO is in the early phase, additional points have to be investigated. Firstly, the three approaches have to be further investigate. For a full support of QoS modeling in WSMO the third approach seams the most promising. Secondly, a clear syntax for constructs supporting the third approach has to be defined. Finally, formal semantics has to be defined for the constructs introduced in the language, which will enable reasoning on QoS characteristics of services and thus the provision of a certain degree of automation for all service related tasks that consider QoS characteristics (e.g. selection, negotiation, ranking).

7. ACKNOWLEDGMENTS

This work is partially funded by the European Commission under the projects ASG, DIP, enIRaF, InfraWebs, Knowledge Web, Musing, Salero, SEKT, Seemp, SemanticGOV, Super, SWING and TripCom; by Science Foundation Ireland under the DERI-Lion Grant No.SFI/02/CE1/I13; by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects Grisino, RW^2, SemNetMan, SeNSE and TSC. The editors would like to thank to all the members of the WSMO and WSML working groups for their advice and input to this document.

8. ADDITIONAL AUTHORS

Additional authors: Dumitru Roman (DERI Innsbruck, email: dumitru.roman@deri.org) and Thomas Strang (DERI Innsbruck, email: thomas.strang@deri.org) and Dieter Fensel (DERI Innsbruck, email: dieter.fensel@deri.org).

9. REFERENCES

[1] A. S. Ali, O. F. Rana, R. Al-Ali, and D. W. Walker. UDDIe: An Extended Registry for Web Services. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, page 85, Orlando, Florida, USA, January 2003. IEEE Press.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] L. Chung. Non-Functional Requirements for Information Systems Design. In *Proceedings of the 3rd International Conference on Advanced Information Systems Engineering - CAiSE'91, April 7-11, 1991 Trodheim, Norway*, LNCS, pages 5–30. Springer-Verlag, 1991.

[4] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Technical report, DERI, 2005. WSML Final Draft D16.1v0.21. http://www.wsmo.org/TR/d16/d16.1/v0.21/.

[5] C. B. et al. Web Service Execution Environment (WSMX). Technical report, W3C, 2005. W3C Member Submission, http://www.w3.org/Submission/WSMX/.

[6] D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[7] S. Frølund and J. Koistinen. Quality of Service Specification in Distributed Object Systems Design. *Distributed Systems Engineering Journal*, 5(4), December 1998.

[8] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[9] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. In *Proceedings of International Conference on Web Services (ICWS 2005), 2005, Orlando, Florida, USA.*, 2005.

[10] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.

[11] J. W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.

[12] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, January 2003.

[13] E. M. Maximilien and M. P. Singh. A Framework and Ontology for Dynamic Web Services Selection. In *IEEE Internet Computing*, pages 84–93. IEEE Press, September-October 2004.

[14] D. A. Menasce. QoS Issues in Web Services. In *IEEE Internet Computing*, pages 72–75. IEEE Press, November-December 2002.

[15] M. N.Huhns and M. P.Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, January and February:75–81, 2005.

[16] J. O'Sullivan, D. Edmond, and A. H. ter Hofstede. Formal description of non-functional service properties. Technical report, Queensland University of Technology, Brisbane, 2005. Available from http://www.service-description.com/.

[17] C. Patel, K. Supekar, and Y. Lee. Provisioning Resilient, Adaptive Web Services-based Workflow: A Semantic Modeling Approach. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 480–487, San Diego, California, USA, July 2004. IEEE CS Press.

[18] D. Roman, U. Keller, H. Lausen, R. L. J. de Bruijn, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[19] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical report, The DARPA Agent Markup Language (DAML) Program, http://www.daml.org/services/, 2004.

[20] I. Toma and D. Foxvog. Non-functional properties in Web services. Working draft, Digital Enterprise Research Insitute (DERI), August 2006. Available from `http://www.wsmo.org/TR/d28/d28.4/v0.1/`.

[21] V. Tosic, K. Patel, and B. Pagurek. WSOL – Web Service Offerings Language. In *Proceedings of the Workshop on Web Services, e-Business, and the Semantic Web - WES (at CAiSE'02)*, volume 2512 of *LNCS*, pages 57–67, Toronto, Canada, May 2002. Springer Press.

[22] UDDI Spec Technical Committee. UDDI Version 3.0.1. http://uddi.org/pubs/uddi-v3.0.1-20031014.pdf, 2003.

[23] G. Wang, A. Chen, C. Wang, C. Fung, and S. Uczekaj. Integrated Quality of Service (QoS) Management in Service-Oriented Enterprise Architectures. In *Proceedings of the 8th International Enterprise Distributed Object Computing Conference (EDOC'04)*, pages 21–32, Monterey, California, USA, September 2004. IEEE Press.

[24] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. RFC 2413 - Dublin Core Metadata for Resource Discovery. Technical report, Internet Engineering Task Force (IETF), 1998.

[25] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Transactions*, 30(5):311–327, May 2004.