2009 IEEE International Conference on Semantic Computing

Stefan Schulte, Nicolas Repp, Dieter Schuller, Ralf Steinmetz:
From Web Service Policies to Automatic Deviation Handling: Supporting Semantic Description of Reactions to Policy Violations. In: Third IEEE International Conference on Semantic Computing (ICSC 2009), Berkeley, CA, USA - September 14-16, 2009, September 2009.

From Web Service Policies to Automatic Deviation Handling: Supporting Semantic Description of Reactions to Policy Violations

Stefan Schulte, Nicolas Repp, Dieter Schuller, Ralf Steinmetz, Fellow, IEEE
Technische Universität Darmstadt
Multimedia Communications Lab
Merckstr. 25, 64283 Darmstadt, Germany
schulte@KOM.tu-darmstadt.de

Abstract

In scenarios where Web services are involved, Quality of Service (QoS) parameters play an important role when it comes to service selection, service compositions, and runtime behavior of the execution environment, e.g. a workflow engine. Required or guaranteed QoS levels are often described by means of Web service policies. However, policy languages do usually not include the handling of deviations respectively reactions. In this paper, we introduce WS-Re2Policy 2.0, a semantic extension of our former work on a policy language which makes use of WS-Policy in order to describe requirements and handling of deviations through defined reactions. Furthermore, we present a basic ontology for reactions which we make use of in order to semantically annotate reactions in WS-Re2Policy 2.0.

1 Introduction

In recent years, Web services and service-oriented architectures (SOA) have been the subject of large interest both in research and industry. SOA is expected to affect the way software is build and organizations collaborate with each other, e.g., by introducing service-based outsourcing of single services or (parts of) business processes. Here, service consumers need to know which kind of functionalities a service offers and which Quality of Service (QoS) level it is able to provide [3, 15]. This is especially the case if considering services which are provided by third parties and hosted on remote computers which are outside of the immediate control of the service consumer.

Usually, functionalities and guaranteed QoS levels are published in a service level agreement (SLA), i.e., the result of a negotiation between service consumer and provider. Based on the information stored in an SLA, it is possible to detect violations and select a particular Web service according to individual preferences, e.g. the overall required QoS level for a Web service composition. Hints on how to handle a deviation from anticipated or required QoS levels is one further step towards the vision of automatic Web service execution, composition and interoperation as described in [14]. Here, a necessary prerequisite is a comprehensible description of both requirements and reactions if a requirement cannot be met by a service. While there is a couple of approaches incorporating the description of requirements in so-called *policies*, the handling of deviations respectively reactions is mostly disregarded (cp. Sections 2.1 and 3).

In this paper we introduce WS-Re2Policy 2.0, a language which makes it possible to combine requirements and reactions. In contrast to its previous versions, it provides on the one hand a semantic description of policies and reactions and on the other hand enhances the underlying work by new constructs (e.g., subsequent reactions) which have not been handled in our previous work. In order to do so, we make use of an existing QoS-ontology and model a basic reactions-ontology/taxonomy.

The remaining part of this paper is structured as follows. In the next section, related work and a brief review of the previous version of WS-Re2Policy is presented. In Section 3 the course of action followed in order to enhance the language by semantic descriptions is presented. Subsequently, the constructs of WS-Re2Policy 2.0 are discussed in more detail and explained by the use of an example. The paper closes with a conclusion and an outlook on our future work.

2 Foundations

2.1 Related work

To the best of our knowledge, there is no standard for Web service-related policies which incorporates requirements *and* reactions. However, apart from our former work (cp. Section 2.2), some researchers have incorporated ba-

978-0-7695-3800-6/09 $26.00 ©
DOI 10.1109/ICSC.2009.70

sic policy enforcement aspects, e.g., Ludwig et al. [13]. Here, the authors make use of WS-Policy as a part of WS-Agreement to specify requirements in their CREMONA architecture. However, the focus of this work is on the initial negotiation of a SLA and possible renegotiations, further reactions are not considered. Tosic et al. [21] also define basic reactions in their *Web Service Offerings Language*, which are restricted to monetary compensation. More advanced definitions of reactions can be found in [2] and [5].

Last but not least, Kagal et al. [11] introduce the policy framework *Rein* with special regard to access control to Web resources. Rein is made up from a number of ontologies which can be used in order to model policies and delegations amongst others but does not provide a language for policies. Delegation is related to security issues and therefore limited to delegation of authority and trust. Nevertheless, it is possible to employ an own policy or reactions language in Rein which makes it possible to model a deviation handling which is not limited to security constraints.

None of the aforementioned work has incorporated semantic information in the context of general policy enforcement. However, as it will be described in Section 3.1, some researchers have tackled the usage of QoS-based semantic information in Web services in general and semantic enhancement of WS-Policy in special.

2.2 Status quo

WS-Re2Policy is a requirements and reactions policy language which has been developed in order to monitor (Web) services in a distributed fashion. The intention of WS-Re2Policy is to meet the requirements of a "classical" scenario for cross-organizational collaborations based on the integration of business processes and by utilization of Web service standards. In such a scenario, customers make use of third party services, i.e. services which are provided by business partners. QoS level guarantees are negotiated a priori to the actual service requests and written down in an SLA. At this point, the definition of reactions, if a QoS requirement is violated, comes into play – if a service is executed remotely (from the perspective of the service consumer), it is necessary to define reactions at the service endpoint.

WS-Re2Policy makes use of the W3C's *Web Services Policy 1.2 framework* (WS-Policy [1]) and is compliant with this framework. Further on, WS-Re2Policy can be extended itself by other WS-Policy compliant languages. Policies in WS-Re2Policy are so-called *RequirementsReactionsSuites*, which are made up from a *requirements* and a *reactions* part. While requirements can be described using any WS-Policy compliant language, reactions are a proprietary development. Possible reactions to SLA violations include amongst others the restarting of a service, renegotiating of SLA parameters, or the selection of a different service. Different types of reactions might be combined or reactions might be repeated.

A more detailed discussion of the previous version of WS-Re2Policy including a detailed example can be found in [17, 18].

2.3 Extensions to WS-Re2Policy

Even though WS-Re2Policy covers the necessary elements needed for the description of requirements and the handling of deviations, there are some potential improvements which have evolved during the usage of the initial versions of the language.

Firstly, since the first version of our policy language, WS-Policy has evolved from W3C Member Submission 1.2 [1] to W3C Recommendation 1.5 [23], hence WS-Re2Policy should be adjusted to this progress.

Furthermore, WS-Re2Policy currently supports sequences of reactions as well as iterations (loops). More control constructs like, e.g., break points (i.e., if the reaction leads to a positive outcome, the remaining reactions will be skipped) or simultaneous reactions are thinkable and would lead to more options when describing reactions.

Finally, several authors have proposed the usage of QoS ontologies for Web services (cp. Section 2.1). QoS facets are the foundation of the requirements defined in WS-Re2Policy and therefore it is reasonable to make use of an already existing QoS-ontology in order to define requirements. However, these ontologies do not involve information regarding possible reactions – this makes it necessary to develop a reactions-ontology.

While some of the former shortcomings mentioned above can be met by an extension of the XML schema of WS-Re2Policy, the involvement of semantic information regarding QoS aspects facilitates completely new options on how to process policies, e.g., by using a reasoner which is able to detect policy inclusion, equivalence, incompatibility, incoherence, and conformance [16]. Thus, we decided to re-engineer WS-Re2Policy as a semantically annotated policy language for Web services.

3 Making use of semantic data in WS-Re2Policy

In this section, we present the course of action followed in order to enhance WS-Re2Policy with semantic information. This includes a consideration of related work regarding semantic annotations for Web service policies in general (Section 3.1), the integration of WS-Policy in SAWSDL (Section 3.2) and an examination of QoS- (Section 3.3) and reactions-ontologies (Section 3.4). Regarding the last-mentioned, we will introduce our own basic taxonomy as

we were not able to find any related work we could have made use of.

3.1 Semantic Annotations for Web Service Policies

While WS-Policy delivers per se only a syntactic description of policies, there has been some work on how to enhance such policies with semantic information.

An approach to enrich WS-Policy with semantic information can be found in [12]. Here, the authors provide a general translation of WS-Policy to OWL and do not focus on QoS aspects.

Verma et al. [24] extend WS-Policy by OWL ontologies. Concepts from the ontologies are directly inserted into policy assertions. Furthermore, the authors provide policy matching functionalities. Roman et al. [19] follow a similar approach in their attempt to align WSMO and WS-Policy. Again, the authors embed the policy ontology in the policy assertion.

Most recently, Chaari et al. [4] have enhanced Web service selection by using a QoS-ontology and WS-Policy. The authors present QoS-based policy matching, a generic QoS-based ontology and an extension of UDDI. However, the authors do not make use of common ontology standards like OWL and are therefore not able to incorporate one of the many efforts towards QoS-ontologies (cp. Section 3.3). This approach leads to the benefit that it is possible to extend policies directly in WS-Policy. Nevertheless, in our opinion it would be more elegant to refer directly to the ontology concepts in order to employ the semantic information in common tools for the handling of (semantic) Web services. Furthermore, the usage of a commonly accepted QoS-ontology would facilitate a higher degree of exchangeability and also reusage of the services which the policies refer to.

Garcia and de Toledo [9] enrich QoS policies for Web services by semantic information based on WSDL and OWL. The authors present their own annotation mechanisms and do not make use of those provided by SAWSDL respectively WSDL-S. Like the aforementioned approaches to semantically annotated policies for Web services, this approach does not incorporate any information regarding reactions. Nevertheless, this approach comes closest to our requirements as it will be presented in the following sections.

Considering these approaches to semantic annotations for policies for Web services, we decided to follow the approach by [4] and enhance the description of WS-Re2Policy-based *RequirementsReactionsSuites* by references to concepts in semantic models.

3.2 WS-Policy and SAWSDL

In this work, we make use of SAWSDL in order to annotate QoS-policies semantically. This approach raises two issues – on the one hand, it is necessary to define an ontology which can be used in this scenario (cp. Sections 3.3 and 3.4) and on the other hand it has to be defined which elements of WS-Policy (respectively WS-Re2Policy) should be annotated.

SAWSDL provides the *modelReference*-attribute [8]. This element can be used in order to model references from elements within WSDL (and furthermore XML schema which is not employed in the work at hand). Even though model references may be used with every element within WSDL, their meaning is only defined for a couple of WSDL-elements and most notably not for the various WS-* specifications, e.g., WS-Policy.

Thus, we define a *modelReference*-attribute for WS-Re2Policy analog to its definition in SAWSDL [8]: A *modelReference* can be used to annotate a policy assertion (i.e., a requirement represented in WS-Policy) or a reaction and provides a reference to a concept or concepts in a semantic model that describes the QoS requirement or the reaction. In our scenario, the semantic model is an ontology (cp. sections below) as it seems reasonable to build on the already existing work in this area.

Concerning the definition of a *modelReference* in the WS-Policy framework, the *modelReference* attribute with a non-empty value introduces a policy assertion parameter which may be used to parameterize the behavior indicated by the assertion [23]. The value of this parameter is a set of URIs taken from the value of the attribute [8].

WS-Re2Policy 2.0 is a description language. Mechanisms to match requirements and actual QoS-parameters and start reactions if necessary have to implemented in corresponding applications which make use of WS-Re2Policy 2.0.

3.3 QoS-Ontologies

Even though there has been some reservation on the availability and enforceability of ontologies in general [10], the usage of ontologies which describe QoS aspects seems to be reasonable. QoS aspects and their interdependencies are a relatively established area where new definitions are not a daily occurrence and hence, a QoS ontology does not have to be altered very often. In addition, there has been significant work on the definition of QoS-ontologies in the field of Web service research (see below) which we can build on.

As we restrict this scenario to QoS-related policies and reactions, we need ontologies which provide concepts to describe QoS aspects as well as reactions. While their have been quite large efforts to build QoS-ontologies, to the best

of our knowledge there is no ontology which we could use to describe possible reactions if a requirement cannot be met. Hence, it was necessary to develop our own reactions-ontology (cp. Section 3.4).

As a result of the different research efforts on the definition and engineering of QoS-ontologies (e.g., [6, 20, 25]), miscellaneous QoS-ontology projects have been initialized but none of them has established itself as *the* state-of-the-art QoS-ontology. Surveys on different efforts can be found in [7, 22].

For our work, an ontology should feature at least the following characteristics: a) be public available, b) apply a commonly accepted ontology language, e.g., a variant of OWL, c) be applicable to (SA)WSDL respectively WS-Policy (as proposed in Section 3.2), and d) define QoS properties. Because the work at hand presents an approach on how to annotate QoS requirements and reactions semantically, the ontology applied can be quite lightweight as long as it possesses the aforementioned different QoS properties. However, if regarding further applications, a more heavyweight ontology which incorporates more information could be helpful.

A very promising approach seems to be the *unified QoS/SLA ontology* proposed in [7]. However, as far as we can judge, this project has not released any ontologies to this day. Hence, we decided to make use of the ontology presented in [6], as the author is involved in the work presented in [7]. The choice for an actual ontology is not vital to the work at hand and is only done for demonstration purposes.

3.4 Reactions-Ontology

It was not the aim of the work at hand to develop a domain-independent and "complete" ontology for reactions. Hence, the following realization is restricted to a base reactions ontology/taxonomy realized in OWL DL[1].

The reactions-ontology represents a minimal set of concepts which can be used to describe reactions, i.e., actions that need to be performed if a QoS requirement cannot be met. The main concepts in the ontology are *Reaction* and its subclasses *AutomaticReaction* and *ManualReaction*. As the name implies, the aforementioned is used to carry out an automatic reaction. Therefore, such a reaction has got a default value. This value could indicate how often the reaction should be conducted or how long the reaction should last. A default value has got the type *unit* which specifies its measuring unit. The unit could be time measurement unit, e.g., time in milliseconds or a counter which indicates how often a reaction should be carried out. Some reactions do not have an associated unit.

In the following section, we are making use of the following instances (*individuals*) of the class *AutomaticReaction*: *Delegate* indicates that the service invocation should be delegated to another service host; *Restart* is used to describe that the service should be restarted; *Sleep* denotes that the invoking application should wait before the next step; *Logging* shows how the reaction should be logged.

4 WS-Re2Policy 2.0 – Constructs and Demonstrator

In the following, we enhance WS-Re2Policy by new control constructs using the ontologies presented in the previous section to annotate requirements and reactions (cp. Section 4.1) and present a demonstrator which makes use of WS-Re2Policy 2.0 in order to implement a distributed service monitoring and SLA enforcement approach (cp. Section 4.2).

4.1 Constructs

An overview on the constructs used in WS-Re2Policy 2.0 can be found in Table 1[2]. For the different kinds of WS-Policy-based policy assertions, we refer to [23].

Reactions are processed in the sequence they are found in the *Reactions*-part of a *RequirementsReactionsSuite* except if it is stated that two or more reactions should be carried out simultaneously. Usually, the processing of reactions stops if a previous reaction has been carried out successfully. However, if there are existing reactions in the sequence which possess the attribute `re2:Usage="required"`, these will be processed in every case while `re2:Usage="optional"` indicates that such a reaction does not have to be carried out if another reaction was successful. Furthermore, if there is a reaction with the attribute `re2:Usage="final"`, it will be carried out as the last reaction independent of its position in the sequence. As it can be seen in Figure 1, this facilitates the modeling of different reaction sequences.

Using the example in combination with the instances taken from the reactions taxonomy from Section 3.4, a Web service execution environment would first wait for 10000 milliseconds before trying to restart the service at most four times (time in milliseconds is the unit attached to *Sleeping* in the reactions taxonomy). If this is not successful, the control over the regarded Web service would be passed to another peer (i.e., a machine providing another instance of the same Web service). This last step will be skipped if the restarting is successful. In the example, the reactions will be logged by sending a mail to an administrator in every case.

[1] `http://www.kom.tu-darmstadt.de/~schulte/reactions20090715.owl`

[2] We assume that `re2` is the prefix of the corresponding XML namespace for WS-Re2Policy 2.0

Construct	Usage/Comment
re2:RequirementsReactionsSuite	A potentially empty set of requirements and reactions which is made up from re2:Requirements and re2:Reactions.
re2:Requirements wsp:Policy qos:XXXXXX	A potentially empty set of requirements (policy assertions). A potentially empty collection of policy assertions. A QoS-related policy assertion. Apart from the usual attributes a policy assertion might have, a QoS-related policy assertion might have a *sawsdl:modelReference*-attribut pointing to a related concept in a QoS-ontology.
re2:Reactions re2:Sleep, re2:RestartService, re2:DelegateControl re2:LogReactions re2:usage	A potentially empty collection of reactions which gives information what needs to be done if a policy assertion cannot be met. Example reactions, might possess the attributes re2:usage and sawsdl:modelReference. The latter points to a related concept in a reactions-ontology. Reactions might have an empty value. If so, the default value (if defined) from the sawsdl:modelReference is taken. Might possess the values required, optional, final.

Table 1. Constructs in WS-Re2Policy 2.0

```
<re2:Reactions RequirementsID="3428">
   <re2:Sleep re2:usage="required"
    sawsdl:modelReference="http://www.kom.tu-darmstadt.de/~schulte/reactions20090715.owl#Sleep">10000
   </re2:Sleep>
   <re2:RestartService re2:usage="required"
    sawsdl:modelReference="http://www.kom.tu-darmstadt.de/~schulte/reactions20090715.owl#Restart">4
   </re2:RestartService>
   <re2:DelegateControl re2:usage="optional"
    sawsdl:modelReference="http://www.kom.tu-darmstadt.de/~schulte/reactions20090715.owll#Delegate">peer
   </re2:DelegateControl>
   <re2:LogReactions re2:usage="final"
    sawsdl:modelReference="http://www.kom.tu-darmstadt.de/~schulte/reactions20090715.owl#Logging">Mail_Admin
   </re2:LogReactions>
</re2:Reactions>
```

Figure 1. XML Snippet: Definition of Reactions in WS-Re2Policy 2.0

4.2 Implementation

As we have stated before, the automatic handling of reactions is one further step towards the vision of automatic composition and execution of Web service-based workflows. In our *Automated Monitoring and Alignment* architecture *AMAS.KOM* [18], WS-Re2Policy 2.0 is used in order to describe QoS requirements and reactions in an agent-based monitoring and deviation handling approach.

In AMAS.KOM, an existing workflow description is analyzed and modified in order to integrate proxy services for the redirection of service calls to the monitoring and alignment infrastructure. Policy documents in a machine-readable format, i.e, WS-Re2Policy 2.0, are needed in order to offer information regarding both requirements and associated reactions.

The actual integration of WS-Re2Policy 2.0 in AMAS.KOM is put into action with Woden4SAWSDL. Here, a *RequirementsReactionsSuite* is a Woden-ExtensionElement which makes it possible to access the sawsdl:modelReference-attribute of the suite and accordingly *Requirements* and *Reactions*.

5 Conclusion and Future Work

In this paper, we have presented WS-Re2Policy 2.0, a language which combines WS-Policy-based policy assertions with information regarding deviation handling. The primary language construct is a *RequirementsReactionsSuite*, i.e., a combination of policy assertions and reactions which applies if a deviation is detected. WS-Re2Policy makes use of elements from SAWSDL in order to annotate (QoS-)requirements and reactions semantically, thus facilitating the automatic handling of deviations.

Furthermore, we have presented a basic taxonomy that describes reactions which need to be carried out manually or by a Web service execution environment.

The work at hand is used in AMAS.KOM, where WS-Re2Policy 2.0 is deployed in order to model requirements and reactions in a scenario where it is not possible for the service consumer to enforce a certain deviation handling at runtime due to security constraints. Apart from this scenario, WS-Re2Policy could be applied in other settings as well, e.g., making use of the defined requirements in Web

service compositions in order to derive QoS requirements of a complete workflow out of requirements defined for single services. In general, the language can be applied in every context where it is necessary to handle SLA violations (semi-)automatically.

The application of WS-Re2Policy in other environments will be part of our future work. In addition, we want to extend the prototypical reactions-ontology presented in this paper.

Acknowledgements

This work is supported in part by the BMBF-sponsored project SoKNOS (`www.soknos.de`) and the E-Finance Lab e. V. (`www.efinancelab.de`).

References

[1] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, H. Prafullchandra, C. von Riegen, D. Roth, J. Schlimmer, C. Sharp, J. Shewchuk, A. Vedamuthu, U. Yalçinalp, and D. Orchard. Web services policy 1.2 – framework (WS-Policy), April 2006. W3C Member Submission.

[2] F. Baligand, N. Rivierre, and T. Ledoux. A declarative approach for QoS-aware web service compositions. In *Proceedings of the 5th International Conference on Service-Oriented Computing*, pages 422–428, Berlin, Heidelberg, 2007. Springer.

[3] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Journal of Web Semantics*, 1(3):281–308, 2004.

[4] S. Chaari, Y. Badr, and F. Biennier. Enhancing web service selection by QoS-based ontology and WS-policy. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 2426–2431, New York, NY, USA, 2008. ACM.

[5] D. Comes, S. Bleul, and M. Zapf. Management of business processes with the bprules language in service oriented computing. In *Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009*, Kassel, Germany, 2009.

[6] G. Dobson, S. Hall, and G. Kotonya. A domain-independent ontology for non-functional requirements. In *IEEE International Conference on E-Business Engineering (ICEBE'07)*, pages 563–566, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[7] G. Dobson and A. Sánchez-Macián. Towards unified QoS/SLA ontologies. In *Proceedings of the 2006 IEEE Services Computing Workshops (SCW 2006)*, pages 169–174, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[8] J. Farrell and H. Lausen. Semantic annotations for WSDL and XML schema, August 2007. W3C Recommendation.

[9] D. Z. G. Garcia and M. B. F. de Toledo. Semantics-enriched QoS policies for web service interactions. In *WebMedia'06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, pages 35–44, New York, USA, 2006. ACM.

[10] M. Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing*, 11(1):90–96, 2007.

[11] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Self-describing delegation networks for the web. In *IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 205–214, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[12] V. Kolovski and B. Parsia. WS-policy and beyond: application of OWL defaults to Web service policies. In *2nd International Semantic Web Policy Workshop (SPSW'06)*, 2006.

[13] H. Ludwig, A. Dan, and R. Kearney. Cremona: An architecture and library for creation and monitoring of WS-agreements. In *Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 65–74, New York, NY, USA, 2004. ACM.

[14] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[15] D. A. Menascé. QoS issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[16] B. Parsia, V. Kolovski, and J. Hendler. Expressing WS Policies in OWL. In *Policy Management for the Web*, 2005.

[17] N. Repp, J. Eckert, S. Schulte, M. Niemann, R. Berbner, and R. Steinmetz. Towards automated monitoring and alignment of service-based workflows. In *IEEE International Conference on Digital Ecosystems and Technologies 2008 (IEEE DEST 2008), Phitsanulok, Thailand*, 2008.

[18] N. Repp, A. Miede, M. Niemann, and R. Steinmetz. WS-Re2Policy: A policy language for distributed sla monitoring and enforcement. In *Proceedings of the Third International Conference on Systems and Networks Communications*, pages 256–261. IEEE Press, 2008.

[19] D. Roman, J. Kopecky, I. Toma, and D. Fensel. Aligning WSMO and WS-Policy. In *2nd International Semantic Web Policy Workshop (SWPW'06)*, 2006.

[20] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. S. Freie. A concept for QoS integration in web services. In *1st Web Services Quality Workshop (WQW 2003)*, pages 149–155, 2003.

[21] V. Tosic, B. Pagurek, and K. Patel. WSOL – a language for the formal specification of various constraints and classes of service for web services. In *International Conference on Web Services 2003, Las Vegas, US*, pages 375–381, 2003.

[22] V. X. Tran and H. Tsuji. Semantics in QoS for web services: A survey, 2008. http://sigswo.org/papers/SIG-SWO-A801/SIG-SWO-A801-02.pdf, last access at 2009-04-17.

[23] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and U. Yalçınalp. Web services policy 1.5 – framework, September 2007. W3C Recommendation.

[24] K. Verma, R. Akkiraju, and R. Goodwin. Semantic matching of web service policies. In *Second International Workshop on Semantic and Dynamic Web Processes (SDWP 2005)*, pages 79–90, 2005.

[25] C. Zhou, L.-T. Chia, and B.-S. Lee. DAML-QoS ontology for web services. In *IEEE International Conference on Web Services (ICWS 2004)*, pages 472–479, 2004.