QoS awareness support in Web-Service semantics

CONFERENCE PAPER · MARCH 2006

DOI: 10.1109/AICT-ICIW.2006.156 · Source: IEEE Xplore

4 AUTHORS, INCLUDING:

Ioanna Roussaki
National Technical University of Athens
86 PUBLICATIONS **621** CITATIONS
SEE PROFILE

Ioannis V. Papaioannou
National Technical University of Athens
35 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Miltiades Anagnostou
National Technical University of Athens
145 PUBLICATIONS **860** CITATIONS
SEE PROFILE

QoS awareness support in Web-Service semantics

Dimitrios T. Tsesmetzis, Ioanna G. Roussaki, Ioannis V. Papaioannou, Miltiades E. Anagnostou
School of Electrical and Computer Engineering, National Technical University of Athens, Greece
{dtsesme,nanario,jpapai,miltos}@telecom.ntua.gr

Abstract

Web Services (WSs) are a new breed of web application that have brought out quite challenging research issues. One of these is the establishment of an interoperable semantic framework suitable to represent all potential features of WSs. Apart from the functional properties that have already been modeled via standardized tools, there are also the non-functional features of WSs i.e. their Quality-of-Service, which in most cases are not included in the WS description. Nevertheless, integrating QoS features in WS profiles is to the advantage of both users and providers, as it supports QoS-aware WS selection and composition addressing the user's QoS requirements, while enabling WS providers to increase their profit in the e-business domain. This paper is concerned with the creation of a QoS ontology framework adequate for WS provision. It has sprang from the work carried out in the IST-Amigo[1] Integrated Project for Ambient Intelligence (AmI) homes, which aims to develop an open, standardized platform for the provision of QoS-aware AmI Web Services in home environments.

1. Introduction

Web Services (WSs) [1] are considered to be the Web's next revolution and the future of e-business. They appeared just 5 years ago and have been one of the most popular research fields ever since. These services are "*self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be anything from simple requests to complicated business processes. … Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.*" [2]. Considerable work has been made to develop adequate middleware for the deployment of Web Services. Towards this end, various technologies have been used such as [3]: XML[2] (Extensible Markup Language), SOAP[3] (Simple object Access Protocol), UDDI[4] (Universal Discovery, Description and Integration), WSDL[5] (Web Services Description Language), SOA (Service-Oriented Architecture) [4], etc. Some of the relevant research efforts have focused on developing ontologies that capture the WSs' main properties. Nevertheless, little work has been done to represent the non-functional features of WSs [5], the most critical part of which concerns their Quality of Service (QoS).

Both users and providers can benefit greatly from the incorporation of QoS features in WS profiles. On the one hand, QoS profiles of WSs are crucial in determining which service best addresses the user needs. They have the potential to optimize the user's WS-experience regarding features such as performance, reliability, security, integrity, and cost, in case the user's QoS preferences are considered by the WS selection and composition. On the other hand, QoS can give WS providers a significant competitive advantage in the e-business domain, as QoS-aware services meet user needs better and thus attract more customers. Adopting a WS best-effort policy that does not provide any guarantees on response time, security, throughput, or availability, may still be acceptable in simple WSs. Nevertheless, such policies are totally unacceptable in more demanding cases [6], when for instance dynamic composition of heterogeneous WSs provided by different administrative domains is required. Moreover, WS providers can increase their profits, as considering the QoS profiles in dynamic

[1] This work has in part been supported by the European Union project "Amigo-Ambient intelligence for the networked home environment". More information on www.amigo-project.org.

[2] http://www.w3.org/XML/
[3] http://www.w3.org/TR/soap/
[4] http://www.uddi.org/
[5] http://www.w3.org/TR/wsdl

network resource allocation mechanisms enables them to maximize the utilization of their infrastructure [7].

As the advantages of QoS featured WS profiles became evident, some research initiatives focused on building QoS semantics and ontology management schemes for Web Services. Nevertheless, they mainly developed QoS ontology vocabularies, identifying the various QoS parameters that are involved in Web Service provision [8][9]. However, as QoS parameters can be a lot more than type-value pairs, the need to develop a flexible, highly descriptive and widely applicable solution for the representation of various heterogeneous QoS parameters in a machine-interpretable manner, while supporting enhanced reasoning functionalities, has risen. In this paper a QoS ontology language is introduced that provides a standard model to formally describe arbitrary QoS parameters. It establishes a set of rules, classes, object-properties and data-properties that are used to represent QoS parameters along with the relationships among them. It has the potential to incorporate the plethora of information concerning each QoS feature of WSs and exhibits properties such as interoperability, completeness, flexibility, scalability reliability, and accuracy. Furthermore, a QoS ontology vocabulary is provided where the various QoS concepts involved in WS provision are identified. Thus, a robust QoS semantic framework for WSs is built based on the QoS ontology language and vocabulary developed, enabling providers to increase their gains and users to enjoy intelligent QoS-aware WSs.

The rest of the paper is structured as follows. Section 2 introduces the research work on QoS ontologies. Section 3 presents the QoS ontology language designed for the needs of Web Services. In section 4 the QoS ontology vocabulary identified is briefly described. In section 5, the QoS ontologies developed are instantiated for the "*MultimediaTelemedicine*" Web Service example. Finally, section 6 draws the paper conclusions and elaborates on our future plans.

2. Literature Review

Studying the QoS ontology literature one may see that research has focused on the design of QoS Ontologies to be used in the discovery mechanisms of the published web services. Nevertheless, most of these ontologies are domain specific and not balanced enough to be adequate for most WSs.

The framework presented in [5] is based on agents. Service providers publish their services to registries and agencies, and service consumers use their agents in order to discover the desired service. The designed ontologies (language and vocabulary) enable agent-based dynamic Web Services selection. Although the proposed ontology language is quite complete, the metrics concept is absent. On the other hand, work in [10] has focused on Metrics, Measurement Units, Currencies and other ontologies, necessary for the QoS management. [10] identifies the requirements of Metrics ontologies and studies the relationships and the dependencies between various QoS metrics. A disadvantage of this approach is that even though the provided analysis is quite detailed, it is limited only to the aforementioned ontologies, not providing an integrated solution.

[11], [12] and [13] have focused on integrating the QoS awareness functionality in the WS discovery mechanisms. In [11] the defined QoS ontology language consists of three layers: the QoS profile layer, which is used for matchmaking purposes; the QoS property definition layer, which is used for elaborating the property's domain and range constraints; and the metrics layer that provides measurement details. A drawback of this modeling approach is that the proposed ontology is quite limited, while the QoS ontology vocabulary is absent. [12] has focused on extending the matchmaking mechanisms with the concept of the service broker. An advantage of this approach is that it enables users to monitor the status of the server. The proposed QoS ontology classifies the QoS parameters provided in two main categories: network- and server-client related parameters. Nevertheless, it does not provide an advanced QoS ontology, but only a simple XML Schema. A more sophisticated approach based on a mathematical analysis is adopted in [13]. Effort has been made to extend the service publication and discovery mechanisms with QoS features. To achieve this, they exploit the finite automata theory in multidimensional spaces. A disadvantage of this approach is that the proposed model performs well when dealing only with static QoS parameters.

[9] does not provide a QoS ontology language or vocabulary specification. However, it elaborates on a classification of QoS parameters, which can be quite useful when attempting to design a QoS ontology vocabulary. Finally, the MOQ (Mid-Level Ontologies for Quality) framework presented in [14] aims to minimize ambiguities in QoS evaluations. Four Ontologies are defined: Requirements, Measurement, Traceability and Quality Management. A drawback of this approach is that it resembles an add-in to existing ontologies rather than a complete QoS ontology.

Our work has focused on the design of a QoS ontology language and vocabulary adequate for arbitrary Web Services. These ontologies are quite detailed, complete and flexible, enabling the efficient

representation of WSs even for the most complicated and demanding cases.

3. A QoS Ontology Language for Web Services

The designed QoS ontology language provides a standard generic model for arbitrary QoS attributes, while defining the nature of associations between QoS attributes and the way they are measured. In our QoS ontology, each QoS attribute is described by the following classes:

- *QoSParameter.* QoS parameter represents a non functional property of the service within a specific domain. These properties may be measurable or not and may hold relationships to each other.
- *Metric.* This class defines the way each QoS parameter is assigned with a value. It is associated with the QoSParameter class through the hasMetric object property. Each Metric object consists of a MetricType and a Value, which are modeled as datatype properties having xsd:string values. The MetricType datatype property is an enumerated string (xsd:enumeration) that represents the QoSParameter's data type, e.g., int, long, string, boolean, etc. Value is a datatype property that formulates the QoSParameter's value as a string. Together with the MetricType property, the system can easily extract the semantics of this information. The Metric class is also related with the Unit class via the hasUnit object property that defines the units used to measure the QoS parameter's quantity. Of course each QoS parameter can either be measurable or unmeasurable. In the latter case, the Unit is set to null. As there are various ways to express a physical quantity in terms of units, the Unit class holds a relationship with the ConversionFormula class that is introduced to enable the transformation from one unit to another. Thus, each Unit object is related to a ConversionFormula, while the ConversionFormula class holds a convertsTo object property having range another Unit object. The QoS ontology also supports statistical analysis elements over the monitored QoS parameters. This functionality is provided by the Statistics subclass of Metric that includes various statistical functions.
- *QoSImpact*. The QoSImpact object property represents the way the QoSParameter value contributes to the service quality perceived by the user. For instance, a reduction on the service latency is expected to increase the quality utility for the user. The QoSImpact property enables the system to estimate the degree of user satisfaction with regards to a given QoS parameter value.
- *Type*. The Type class represents the specific QoS category of the QoS ontology vocabulary, where the QoSParameter belongs to (e.g. "Jitter", "Cost",). It is associated with the QoSParameter class through the hasType object property.
- *Nature*. This datatype property of the QoS parameter represents its static or dynamic nature. A QoSParameter that is defined apriori and does not change during the entire service session is a Static QoSParameter, e.g. the security protocols supported by the service. On the other hand, QoSParameters that may vary during the service execution time are Dynamic, e.g. the service response time. The values of the Nature datatype property are defined by the Service Provider and are periodically confirmed in the user domain. The Nature property is formulated as enumerated string, with range values: "Static" and "Dynamic".
- *Aggregated*. The QoSParameter that is composed by two or more defined QoSParameters has the object property of aggregation. For example, the service response time is composed by the latency and the request process time by the server.
- *Node*. The Node datatype property of the QoSParameter identifies the network node that may have an impact on its value. Thus, each QoSParameter may depend on the Server node attributes, the Client node attributes or both. It is formulated as an enumerated string with range values: "Client" and "Server".
- *Relationship*. This class represents the way a QoSParameter is correlated with others. It is related to the QoSParameter class via a hasRelationship optional object property. In order to interrelate two QoSParameter objects the infuentialParameter mandatory object property has been introduced, which indicates (i.e. has range) the QoSParameter that has an impact on the "owner" QoSParameter of the Relationship. This approach may also handle the case of asymmetric interdependencies between QoS parameters. The Relationship may be Proportional or InverselyProportional. This feature is modeled by the IFType datatype property that is an enumerated string with range values: "Proportional" and "InverselyProportional". For example, the service response time and the throughput are InverselyProportional parameters. The Relationship may also be Strong, Medium or Weak. This information is captured by the ValidityLevel datatype property that also has an xsd:string (xsd:enumeration) value range. The ImpactFactor class is introduced to encapsulate the

two properties above (i.e. IFType & ValidityLevel) that characterize the Relationship. The hasImpactFactor object property is used to bind a Relationship to an ImpactFactor object.

The designed QoS ontology language is presented in Figure 1.

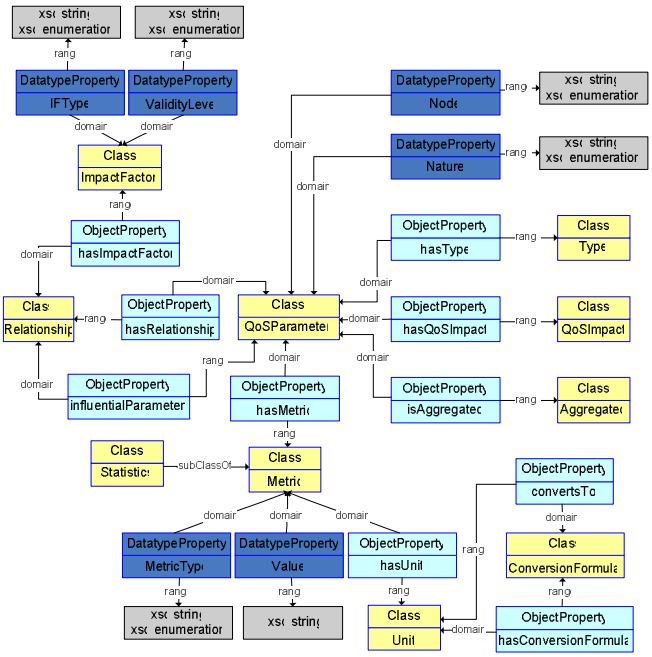

Figure 1. The QoS ontology language

4. The QoS Ontology Vocabulary

An extended review of the literature on QoS taxonomies and classifications (e.g., [8] [9]) coupled with the WSs' requirements formed the basis of our work towards the design of a QoS ontology vocabulary for WSs. In this ontology, all QoS parameters are instantiated as subclasses of the QoSParameter class. Subsequently, an overview of the parameters identified in the QoS vocabulary is provided.

- *Performance*. The Performance QoS parameter aggregates information that mainly depends on the properties of the network connection between the user and the service provider nodes. The subclasses of the Performance class are: *Throughput*, *Latency*, *ResponseTime*, *Jitter* and *ErrorRate*. In general, high performance services should provide high throughput, low latency, fast response time, low jitter and low error rate.
- *Scalability*. The Scalability QoS parameter is used to describe the server's ability to increase its computing capacity and the system's ability to process more users' requests, operations or transactions in a given time interval.
- *Availability*. Availability is defined as the probability that the server is up and running. Availability varies between 0 and 1. When it is closer to 1, the server is considered to be highly available, a feature always necessary.
- *Accessibility*. Accessibility refers to server's ability of serving a Web Service request. Accessibility is different from availability, as a service maybe available, i.e. the server is up, but can not handle any incoming requests, thus not being accessible.
- *Accuracy*. This QoS parameter refers to the accuracy of results in a numerical manner. It specifies the number of significant decimal digits of results.
- *Capacity*. Capacity specifies the maximum number of requests a server is able to handle simultaneously.
- *Cost*. This QoS parameter represents the overall cost that results from service usage.
- *Configuration*. The configuration of services is related to the interface update procedure and/or the adopted standards; and it provides information about the regulations the service complies with. It indicates whether the services can interact with each other. Configuration is measured by the following metrics: *Stability* that represents how often the service interfaces are modified, *SupportedStandard* that refers to the standards that the service complies with, and *Regulatory* that refers to the probability of the fact that the service is compliant with a random regulation.
- *Integrity*. It represents the ability of a WS to preserve data integrity during a transaction. In order to accomplish data integrity, all the transactions that implement a specified function are handled as a single unit. In case the transaction is completed successfully all changes in data are committed, while if the transaction is not completed, all changes are rolled back. Integrity of data is a boolean QoS parameter that is either supported by the service or not.
- *Reliability*. This QoS parameter represents the possibility of a WS session to get completed successfully. Reliability is closely related to availability, as the more available a server is, the more reliable its services are. The parameters that measure reliability are: *MTTR* (Mean Time To Repair) and *MTBF* (Mean Time Between Failures).
- *Security*. This category of QoS parameters refers to the security level a service provides. Security is of great importance in AmI environments, as the user needs to be in control, while the cognitive saturation of the user should be avoided and user privacy should be protected. The security

subclasses defined are: *Confidentiality*, *Auditability*, *Authentication*, *Authorization*, *DataEncryption* and *NonRepudiation*.

The main classes of this QoS ontology vocabulary are depicted in Figure 2

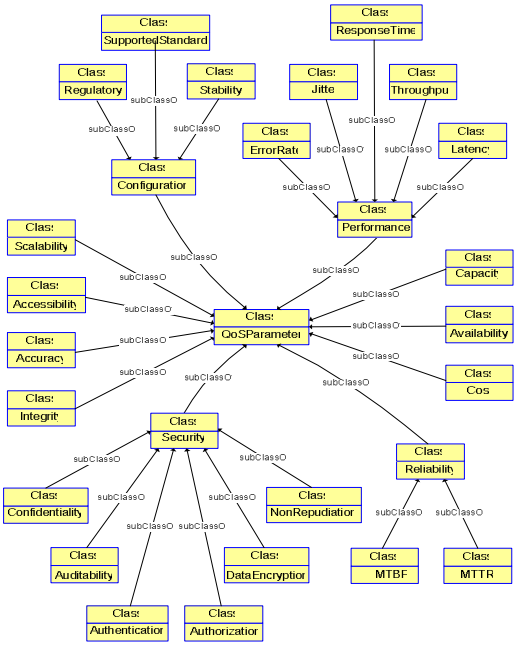

Figure 2. The QoS ontology vocabulary.

It should be mentioned that the maximum height of the defined QoS taxonomy tree is two. This is preferable for the development of a QoS managements system demonstrating reduced complexity.

Both the QoS ontology language and vocabulary have been implemented in OWL using the Protégé [15] ontology editor. The specified QoS vocabulary and language are used in the Amigo Interoperable Middleware in order to represent the QoS of the interacting services. These ontologies form the basis of the QoS profiles defined for the WSs delivered to the Amigo users. OWL-S[6] has been used to integrate them with existing WSs in order for the latter to become QoS-aware. The parameters of these ontologies are managed via a module involved in QoS-aware service management in Amigo. A reasoning mechanism is necessary to ensure consistency of the different service instantiations and the various vocabularies used by the parties involved. Several technologies are currently being examined as ontology reasoners and inference engines candidates. Among these, the most likely to be used is the Jena 2 semantic web framework[7] with an external plugged-in reasoner (probably Racer[8]). Our QoS ontology's instances should be discoverable. In order to support this, various discovery protocols will be used, such as the WS-Discovery[9], the UPnP[10] and the RMI[11]. In March 2006, a prototype implementation of the Amigo Ambient Intelligence home platform will be finalized based on the OSGi framework[12].

5. A QoS Ontology Instantiation Use Case

In order to illustrate the functionality and usefulness of the ontologies developed, an instantiation of the designed QoS ontology is provided in this section. The use case selected concerns WS provision in the ambient intelligence home domain. In this respect, the QoS ontology instantiation is built for the following scene [16]:

"Maria is an elder person who needs to have frequent contact with her doctor. Using its telemedicine set box, Maria can have these contacts periodically, previously scheduled or at any moment, when she feels the need for such professional attention. The contact between the doctor/nurse and Maria is facilitated though a telematic media so that to provide a Virtual Person-to-Person interaction, permitting the exchange of images, sound and data (virtual presence). Maria can be alone at home or assisted by relatives or care givers. The doctor who provides the medical attention may work in a hospital/health care centre or may provide his/her professional service from home or a private consultation."

The main service in the scene described above is the "*MultimediaTelemedicine*" service. Key elements are the database holding Maria's medical history, the videoconference session support, and the biomedical data acquisition and transmission functionality. The system supports various measurements ad-hoc transmission, e.g. ECG (electrocardiogram), chest sound, blood pressure, temperature. Additionally, traditional medical devices coupled with biosensors attached to the patient's body are established. They are used to detect medical emergency situations. In such cases the system automatically contacts a doctor, while no action is required from Maria.

The MultimediaTelemedicine service involves a plethora of QoS parameters and requirements that should be addressed in order to ensure an effective and

[6] http://www.daml.org/services/owl-s/1.1/
[7] http://jena.sourceforge.net/
[8] http://www.racer-systems.com/products/racerpro/index.phtml
[9] http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-discovery1004.pdf
[10] http://www.upnp.org/
[11] http://java.sun.com/docs/books/tutorial/rmi/
[12] http://www.osgi.org/

reliable communication between both participants (doctor-patient). The most critical QoS parameters in this scene are those related to network performance, as real-time video communication is involved. Thus, high Throughput should be guaranteed, while Jitter, ResponseTime, Latency and ErrorRate should be minimal. Indicative values for these parameters are given in Table 1. Moreover, the MultimediaTelemedicine service should be Accessible and Available as long as possible, addressing the user's need for immediate contact with medical stuff in any case of emergency. Scalability is also important in this WS as the server should be capable of providing additional resources whenever required, while its Capacity should lie above a specific threshold. The Cost QoS parameter of this service expresses how much the service provider charges the user and what charging scheme it uses (e.g. fixed cost per contact). Another essential parameter is SupportedStandard that indicates the various standards supported by the service and can be used to identify possible incompatibilities among the service requirements and the client features. Finally, high Reliability should also be ensured. Finally, security parameters should be considered in this analysis, as we are dealing with sensitive personal medical information that should not be disclosed to any party apart from the authorised medical stuff.

Based on this analysis, we may now instantiate a QoSParameter profile for the MultimediaTelemedicine service, using the designed QoS ontologies.

Table 1. MultimediaTelemedicine QoSParameters.

Accessibility	0.99000
Availability	0.99995
Performance	
(max) Jitter	1 (msec)
(max) ErrorRate	10^{-5}
(max) Latency	300 (msec)
(min) Throughput	384 (Kbps)
(max) ResponseTime	0.01 (sec)
Cost	3.00 (Euro/minute)
Capacity	200
Scalabillity	0.80
Configuration	
SupportedStandards	"UDDI 3.0"
SupportedStandards	"WSDL 1.1"
Reliability	
MTBF	36,000,000 (sec)
MTTR	1,800 (sec)
Security	
Audiability	1
Authentication	"Password"
Authorization	"SSL"
Confidentiality	1
DataEncryption	"AES-128"
NonRepudiation	1

Notice that only the type, value and units of the QoS parameters are provided above. The following code snippet is a part of the ontology description of the Throughput QoS parameter in OWL based on the language presented.

```
<QoSParameter rdf:ID="Throughput">
<hasQoSImpact>
  <QoSImpact rdf:ID="HighImpact" />
</hasQoSImpact>
<Node rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Server</Node>
<hasType>
  <Type rdf:ID="Performance" />
</hasType>
<Nature rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                                                    Dynamic</Nature>
 <hasRelationship>
  <Relationship rdf:ID="ThroughputvsErrorRate">
   <influentialParameter>
    <QoSParameter rdf:ID="ErrorRate" />
   </influentialParameter>
   <hasImpactFactor>
    <ImpactFactor rdf:ID="InvStrong">
     <IFType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                                    InverselyProportional</IFType>
     <validityLevel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                                                    Strong</validityLevel>
    </ImpactFactor>
   </hasImpactFactor>
  </Relationship>
 </hasRelationship>
 <hasMetric>
  <Metric rdf:ID="ThroughputMetric">
   <Value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">500000</Value>
   <MetricType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
                                                    Long</MetricType>
   <hasUnit>
    <Unit rdf:ID="ThroughputUnit">
     <hasConversionFormula>
      <ConversionFormula rdf:ID="bpsToKbps">
       <convertsTo rdf:resource="#ThroughputUnit" />
      </ConversionFormula>
     </hasConversionFormula>
    </Unit>
   </hasUnit>
  </Metric>
 </hasMetric>
</QoSParameter>
```

6. Conclusions and Future Plans

It is a fact that integrating QoS featured profiles in Web Service provision may benefit both users and providers. It enables QoS-aware WS selection and composition, thus addressing the quality based user requirements with regards to properties such as integrity, performance, security, cost and accuracy. Furthermore, high QoS for a WS can be advertised in its QoS profile in order to attract more customers, which brings a considerable competitive advantage to WS providers. Nevertheless, before these parties can enjoy the advantages of QoS-aware WSs, a uniform, highly descriptive and generic solution for the representation of the plethora of the involved QoS parameters needs to be designed and developed.

This paper presents a QoS semantic framework that is suitable for Web Service provision and aims to combine the advantages of existing QoS representation schemes. On the one hand, it is concerned with the creation of a QoS ontology language that is used to formally describe arbitrary QoS parameters in a flexible and machine-interpretable manner. On the

other hand, it presents the QoS ontology vocabulary established, where the various QoS concepts involved in WS provision are identified. In March 2006, a prototype implementation of the Amigo Ambient Intelligence home platform will be finalized. It will be based on the presented QoS ontologies and will be validated over a blend of heterogeneous technologies encompassing multi-role domains. Various evaluation criteria will be used with regards to performance, scalability, interoperability, usefulness and marketability. This work is expected to further contribute to the integration of WSs in Ambient Intelligence systems, while it will hopefully make a step towards the introduction of Web Services in the wide market.

Future plans aim to exploit the QoS semantic framework presented focusing on two research areas. The first will address the establishment of matching algorithms that, based on the QoS ontologies introduced, will support dynamic WS selection and composition considering the user QoS preferences and former behavior in the system. The second field concerns the design of a QoS based negotiation framework that will use mobile intelligent agents responsible for preparing bids for and evaluate QoS featured offers on behalf of the parties they represent (i.e. WS providers and WS users). The agents will act based on specific negotiation strategies aiming to obtain the maximum benefit for their owners. These tasks in the research areas above, intend to contribute to the realization of a solid integrated QoS middleware for the deployment of robust and scalable QoS-aware Web Services.

7. References

[1] H.M. Deitel, P.J. Deitel, B. Du Waldt, and L. K. Trees, "Web Services: A Technical Introduction", Publisher: Prentice Hall PTR, August 2002.

[2] D. Tidwell, "Web services: the Web's next revolution", IBM DeveloperWorks, November 2000 (Available via www.ibm.com, last visited September 2005).

[3] S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama, "Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI", Publisher: Pearson Education, December 2001.

[4] D. Krafzig, K. Banke, and D. Slama, "Enterprise SOA: Service – Oriented Architecture Best Practices (The Coad Series)", Publisher: Prentice Hall PTR, November 2004.

[5] E.M. Maximilien, and M.P. Singh, "A Framework and Ontology for Dynamic Web Services Selection", vol. 8, no. 5, September-October 2004, pp. 84-93.

[6] H. Ludwig, "Web Services QoS: External SLAs and Internal Policies: Or, How Do We Deliver What We Promise?", 4th IEEE International Conference Web Information Systems Engineering Workshops (WISEW'03), December 2003.

[7] M. Govern, and S. Weagraff, "Web Services: Provider Threat or Opportunity?", White Paper, Convergys Corporation, September 2003.

[8] A. Mani, and A. Nagarajan, "Understanding Quality of Service for Web Services", IBM developerWorks, January 2002 (Available via www.ibm.com, last visited September 2005).

[9] K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park, "QoS for Web Services: Requirements and Possible Approaches", W3C Working Group Note, November 2003.

[10] V. Tosic, B. Esfandiari, B. Pagurek, and K. Patel, "On Requirements for Ontologies in Management of Web Services", International Workshop on Web Services, E-Business, and the Semantic Web, May 2002. (LNCS, vol. 2512, pp. 237-247)

[11] C. Zhou, L. Chia, and B. Lee, "DAML-QoS Ontology for Web Services", International Conference on Web Services 2004 (ICWS04), San Diego, California, USA, July 2004.

[12] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller, "A Concept for QoS Integration in Web Services", 1st Web Services Quality Workshop (WQW 2003), Rome Italy, December 2003.

[13] D. Bianchini, V. De Antonellis, and M. Melchiori, "QoS in ontology-based service classification and discovery", 3rd International Workshop on Web Semantics (WebS 2004), Saragossa, Spain, August 2004.

[14] H.M. Kim, A. Sengupta, and J. Evermann, "MOQ: Web Services Ontologies for QOS and General Quality Evaluations", European Conference on Information Systems (ECIS 2005), Regensburg, Germany, May 2005.

[15] J.H. Gennari, M.A. Musen, R.W. Fergerson, W.E. Grosso, M. Crubézy, H. Eriksson, N.F. Noy, and S.W. Tu, "The evolution of Protégé: an environment for knowledge-based systems development", International. Journal of Human-Computer Studies, vol. 58, no. 1, January 2003, pp. 89-123.

[16] M. Janse (editor), "Amigo-D1.2: Report on User Requirements: State of the Art, Volume II", Amigo Integrated Project (IST-004182), April 2005.