Implementing an SLA Negotiation Framework

Peer HASSELMEYER[1], Henning MERSCH[3], Bastian KOLLER[2],
H.-N. QUYEN[2], Lutz SCHUBERT[2], Philipp WIEDER[3]
[1]*CCRL, NEC Europe Ltd., Rathausallee 10, 53757 St. Augustin, Germany*
Tel: +49-2241-9252-0, Fax: +49-2241-9252-99,
Email: hasselmeyer@ccrl-nece.de
[2]*High Performance Computing Centre Stuttgart (HLRS), Allmandring 30a, 70550 Stuttgart,, Germany*
Tel: +49-711-685-4275, Fax: +49-711-682-357,
Email: {koller, quyen, schubert}@hlrs.de
[3]*Research Centre Jülich (FZJ), 52452 Jülich, Germany*
Tel: +49-2461-61-2760, Fax: +49-2461-61-6656,
Email: {h.mersch, ph.wieder}@fz-juelich.de

Abstract: Despite promising high value for electronic business, Service Level Agreements have not yet found major uptake in the business world. A major obstacle to adoption is the lack of adequate tools that facilitate and automate SLA establishment on both the consumer as well as the provider side. In this paper, we describe a generic framework for negotiating SLAs that has been designed and implemented by the authors. The components automate large parts of the negotiation process while at the same time letting the user retain control. An application scenario has been realised using our framework showing that the negotiation and establishment of SLAs is a viable solution for electronic contracting and that our framework can facilitate this task.

1. Introduction

Service Level Agreements (SLAs) more and more prove their added value not only as in-house solutions for the IT sector, but also as instruments to manage contracts throughout distributed systems and service ecosystems. As of today, this technology has not yet found major uptake in the business world, despite the concepts behind, which are promising high value for this area. Service providers can make use of SLA technology to advertise and offer their services' capabilities while consumers are able to formalise their service level objectives through SLAs. It is in the interests of both parties to create and operate SLAs with a minimum of human interaction on the one hand, but to negotiate and agree upon legally binding electronic contracts on the other hand. Balancing these objectives is a non-trivial task and our work represents a contribution towards a more automated, business-oriented integration of Service Level Agreements into state-of-the-art distributed systems. With that approach we show that SLAs can be a powerful tool enhancing business capabilities of service providers and customers with at the same time decreasing cost and effort.

2. Objectives

The SLA Lifecycle as described by the TeleManagement Forum [1] can be split up in six different phases, as there are:

1. development of service and service templates,

2. discovery and negotiation of an SLA,
3. service provisioning and deployment,
4. execution of the service,
5. assessment and corrective actions during execution (parallel phase to execution of the service), and
6. termination and decommission of the service.

Within the NextGRID [2] project we designed and implemented a framework that concentrates on phase two of the lifecycle – the set-up of SLAs. It includes the discovery of service provider candidates that provide a certain service level as well as the negotiation with those candidates to reach an agreement (the SLA) on the service level requested and (ultimately) provided.

Our approach is led by two main objectives. First, the customer should not deal with Service Level Agreements in their machine-processable form, but specify business objectives, requirements, and preferences in a preferably "natural" way. This implies that the user interface offers choices like e.g. "gold", "silver", or "bronze" service levels instead of presenting the SLA as an XML instance document to be edited within a text editor. And second, the framework has to support the user in getting an agreed-upon SLA. The framework should therefore handle and automate all tasks surrounding the creation of an SLA, including in particular service discovery and SLA negotiation.

We start the paper off with the description of a business-oriented use case to motivate our research. We then describe in Section 4 the architecture of the system and show how its components facilitate the consumer's job of negotiating SLAs. Chapters 5 and 6 provide technical details of our implementation whereas Chapters 7 and 8 present results and business benefits respectively. The last chapter concludes the paper with an outlook on future work.

3. A Business Use Case

From the different NextGRID business use cases used to gather requirements from, we have chosen a basic (but sufficient) business use case of an online shop owner who wants to secure his business by making use of credit card check services.

With the numbers of fraudulent credit cards in circulation, Bob (our shop owner) decides that his company will need to check any credit card before it is accepted for payment. Bob's company has no means to carry out credit card verifications by himself and Bob realises that he will need to find a service provider that offers credit card check services. This service lets merchants verify the validity of credit card payments. It checks whether the presented card is still valid and whether the amount of payment does not exceed the card's credit disposition limit. The service is offered with different quality levels, which differ in dimensions such as the total number of requests allowed, the average processing time, the throughput, and the availability. Bob decides that a service fully outsourced to and looked after by the service provider would suit his needs best. After searching for appropriate providers and obtaining a list of suitable service offerings, the company enters into negotiations with the providers. For Bob, the number of checks allowed in a certain time period and the cost for these checks are the two main aspects for negotiation (his business level objectives).

4. Methodology

The creation of a Service Level Agreement is a process which includes two main parties: the service provider and the service customer. Other parties may be included in the process depending on the requirements of the scenario in question. In case of the credit card check use case, we found two main phases to set up an SLA: First, suitable providers have to be found (*discovery phase*), and second, the terms of the SLA have to be negotiated and agreed upon (*negotiation phase*). Our framework assists customers and providers during both phases, leaving out only the actual service access execution governed by the SLA since this is to a large extent service-specific and would exceed the scope of our work.

The negotiation framework we designed (cf. Figure 1) is structured according to these phases and comprises three categories of components: provider-side, customer-side, and discovery-related. The service provider components comprise, apart from functions to register its service capabilities, a *Negotiator Service* to negotiate SLAs and an *SLA Template Repository* to store information about the offered services and the quality levels at which they can be provided. An *SLA template* contains a description of a particular quality level a provider is offering to provide a service at. An SLA template is not an SLA yet as it has not been agreed upon. Furthermore, it leaves a number of fields inside the SLA document open. The open fields include in particular contact details of the customer, but can also provide room for negotiation of some or all quality attributes. Which fields are left open depends on the negotiation protocol, the application domain, and the service provider. It is important to consider the relation between templates and services. In our approach, a service could (in theory) have an unlimited number of SLA templates (i.e. quality levels). This allows the service provider to offer its services in a flexible manner. The flexibility enhances his competitiveness in the market.

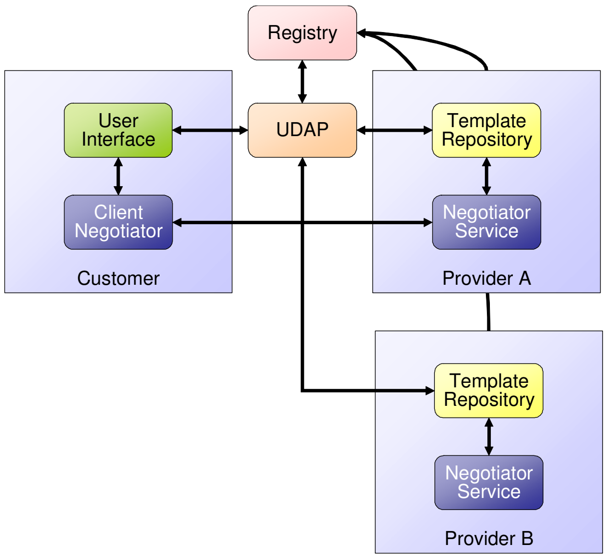


Figure 1: Framework Component Diagram

The Customer combines the *User Interface* and *Client Negotiator*. In between sits the *Universal Dynamic Activity Package* (UDAP), a component which maps the customer's requirements onto the service provider's capabilities. This task is executed making use of the NextGRID service *Registry*, in which all service providers can register their services to advertise them. It is worth mentioning that we foresee that in "real" business scenarios not all available services are registered in a public registry. Based on the business level objectives of the respective service providers it is often in their interest to offer special services to special customers.

The negotiation of an SLA has to follow a certain protocol. We decided to use the so-called *Discrete-Offer-Protocol* which can also be seen as a one-phase negotiation. Using this protocol, the customer sends a request for an offer (called *Bid*) to the service providers' Negotiator Services. This Bid is based on the information received during the discovery phase. The provider itself checks in the SLA Template Repository whether he has one or more matching templates for the requested service and decides on the offer to send to the customer's Client Negotiator (assuming an SLA template exists and the provider wants to make an offer). After having received the offer, the customer now has to decide whether to agree or disagree with this offer and has to inform the service provider's Negotiator Service accordingly.

As indicated in Figure 1, we used the implemented framework to set up a number of different service providers which register their capabilities with the Registry. To start with a basic experiment, we limited the differences between the services (apart from the service's name and endpoint) to the throughput, i.e. the number of checks performed per time unit.

5. Technology Description

The design of our architecture follows service-oriented architecture principles. The services implemented are compliant to the Web Services Resource Framework specification [3] standardised by the Organisation for the Advancement of Structured Information Standards (OASIS).

Concerning the Service Level Agreements we follow the NextGRID approach which models SLAs according to business objectives of both customers and service providers [4]. It has to be noted that, although the current implementation uses NextGRID SLAs (in form of XML instance documents), the framework's design allows the usage of other SLA models. The discovery and negotiation processes described here are independent of the SLA representation whereas some of the components (like the UDAP and the Negotiator Service component) need to be adapted according to the changes to the SLA model.

The negotiation protocol used is the one proposed by WS-Agreement [5]. It follows a Discrete-Offer-Protocol message exchange. The negotiation starts with the client requesting SLA templates from candidate service providers. As mentioned before, SLA templates leave a number of fields open. In our example all quality attributes of the service are fixed. Only the contact details of the service consumer need to be filled in.

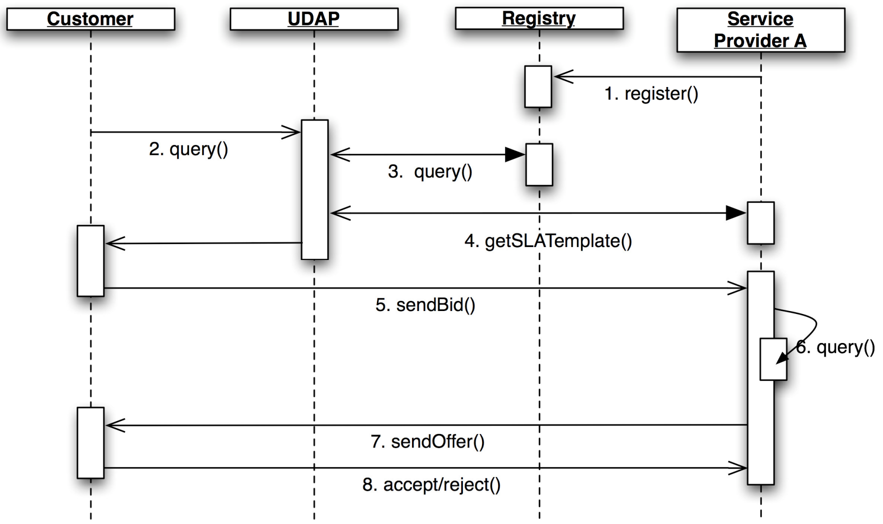


Figure 2: Component Interaction

In the scenario shown in Figure 1, there are two service providers that offer credit card validation services. To simplify the description of component interactions, as shown in the message sequence chart (cf. Figure 2), we only picture Service Provider A. As a prerequisite, the provider registers its service with the Registry (step 1). When the customer sends a query to the UDAP component (step 2) requesting for example "a 'gold' level credit card check service", UDAP executes the respective Registry query (step 3), selects potential candidate services, and sends references to their endpoints together with the matching SLA templates back to the customer. Step 4 is shown here only exemplarily for Service Provider A, but the Registry may return a list with more then one entry if more services are registered. The SLA template is then requested from the provider (which internally gets it from its SLA Template Repository) in step 4 and the result of the query is sent back to the Customer. Subsequently, the Customer contacts all service providers worth being considered by sending a bid to the respective providers (step 5). In case of Figure 2 this is again exemplary shown for Service Provider A. After validating the bid against the templates stored in its repository (for which it is queried in step 6), Service Provider A sends back an offer to the Customer (step 7) to be agreed upon. The Customer may now select the best fitting credit card check service (e.g. the one which offers the largest number of checks per time unit or the one with the least cost). To obtain a valid contract the Customer has to accept the offer (or otherwise reject it) as step 8 indicates.

6. Developments

In the course of our research, we implemented all the components shown in Figure 1. To evaluate interoperability and to demonstrate the openness of our concepts, some of the components are hosted in a Globus Toolkit 4 (GT4) [6] container and some in a UNICORE Version 6 [7] container.

The Registry is a GT4-based service. It realises the Web Services Service Group specification [8] and therefore publishes services in the form of service group entries where each entry represents one of the registered services. Service registrations consist of a service description and a service endpoint at which to contact a service. The service endpoint in our system is the address of the SLA Template Repository. Service discovery is based on the service descriptions which in our demonstrator contain an identifier of the provided service.

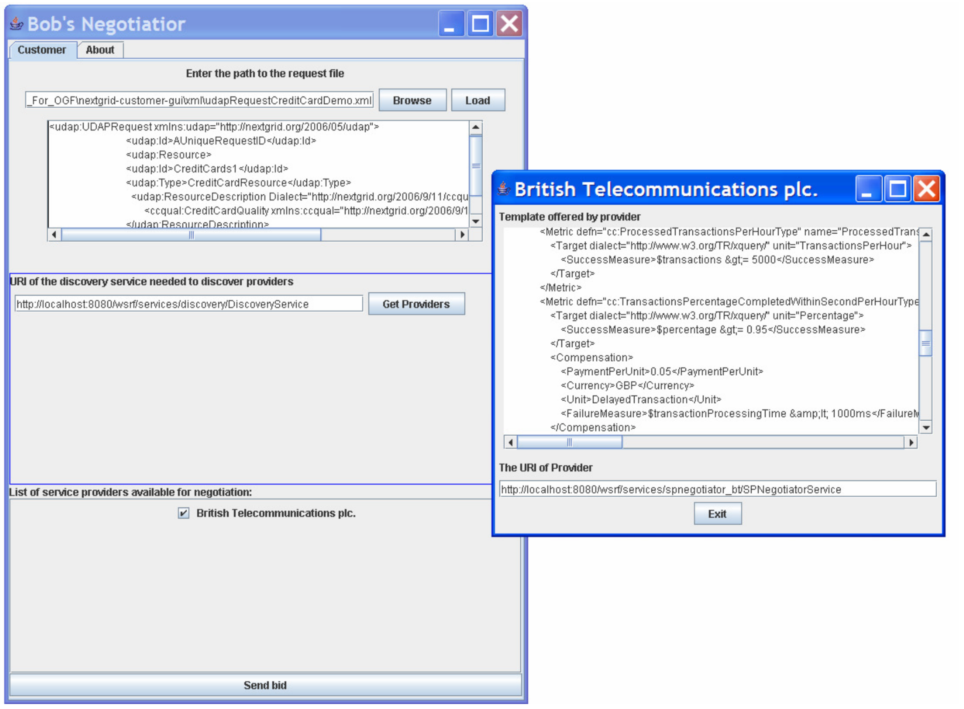
The SLA Template Repository is also a GT4-based service, sitting on the service provider's side with an eXist XML database [9] running in the background. It is used to store and retrieve SLA templates that describe the capabilities of services.

The UDAP framework is a set of services deployed in a UNICORE Version 6 hosting environment. Its main objective is to decouple the customer and provider concerning the requirements descriptions in case of the customer and the capability description respectively. It provides, independent of the type of service requested, service discovery and service matching capabilities.

The Negotiator Service is a GT4-based service, sitting on the service provider side. It is the interface to the outside world and the direct connection to other negotiators during the negotiation phase. When receiving a Bid (i.e. a request for an offer), it starts the mechanism of checking the SLA Template Repository for matching templates. If a match is found and the required resources are available, the Negotiator Service accepts the bid and sends an appropriate offer to the client.

The Client Negotiator is in comparison to the Negotiator Service a less complex service. Its main purpose is to act as a sender/receiver interface to the components outside of the customer's domain. By using the User Interface, requests for discovery, requests for Bids, and acceptances or rejections of offers can be sent to the other parties involved. Based on the information received from the discovery process, the Client Side Negotiator creates the SLA Bids.

The User Interface is a basic GUI (cf. Figure 3), acting as access point for the negotiation on the customer's side. It is used to define locations of discovery request documents and to choose the service provider candidates received through discovery. Additionally, incoming offers are announced through the GUI, can be checked, and finally accepted or rejected. It is important to note that the Graphical User Interface has only been introduced to demonstrate the capabilities of the framework visually. Other applications of the framework already implement service selection algorithms to execute the tasks described before automatically.


Bob's Negotiatior
Customer
About
Enter the path to the request file
Browse
Load
URI of the discovery service needed to discover providers
Get Providers
List of service providers available for negotiation:
Send bid
British Telecommunications plc.
Template offered by provider
The URI of Provider
Exit

_For_OGF\nextgrid-customer-gui\xml\udapRequestCreditCardDemo.xml

```
<udap:UDAPRequest xmlns:udap="http://nextgrid.org/2006/05/udap">
                <udap:Id>AUniqueRequestID</udap:Id>
                <udap:Resource>
                <udap:Id>CreditCards1</udap:Id>
                <udap:Type>CreditCardResource</udap:Type>
                 <udap:ResourceDescription Dialect="http://nextgrid.org/2006/9/11/ccqu
                    <ccqual:CreditCardQuality xmlns:ccqual="http://nextgrid.org/2006/9/1
                </udap:ResourceDescription>
```

http://localhost:8080/wsrf/services/discovery/DiscoveryService

[✓] British Telecommunications plc.

```
<Metric defn="cc:ProcessedTransactionsPerHourType" name="ProcessedTrans
  <Target dialect="http://www.w3.org/TR/xquery/" unit="TransactionsPerHour">
    <SuccessMeasure>$transactions &gt;= 5000</SuccessMeasure>
  </Target>
</Metric>
<Metric defn="cc:TransactionsPercentageCompletedWithinSecondPerHourType
  <Target dialect="http://www.w3.org/TR/xquery/" unit="Percentage">
    <SuccessMeasure>$percentage &gt;= 0.95</SuccessMeasure>
  </Target>
  <Compensation>
    <PaymentPerUnit>0.05</PaymentPerUnit>
    <Currency>GBP</Currency>
    <Unit>DelayedTransaction</Unit>
    <FailureMeasure>$transactionProcessingTime &amp;lt; 1000ms</FailureM
  </Compensation>
```

http://localhost:8080/wsrf/services/spnegotiator_bt/SPNegotiatorService

Figure 3: The Customer Negotiation GUI

7. Results

With the described implementation of the SLA negotiation framework, we were able to retrieve more insight in the usage of SLAs in eBusiness. The automation we achieved with our system covers the following aspects:

- discovery of service providers,
- selection of providers that offer the required service,
- retrieval of service offers (SLA templates),
- visualisation of SLA templates, and
- establishment of an actual SLA.

During the design of the framework and its validation afterwards, it became clear that in general all steps in this phase of the SLA lifecycle can be automated, it depends only on the capabilities of the underlying technology.

Other research projects like, for example, BREIN are currently investigating the usage of multi-agent concepts for automated SLA negotiation. However, when talking about the advantages of automation of these processes it is important to keep in mind that complete automation is not always in the interest of service providers. Decisions must still be guided by human operators. And, in particular when it comes to high-profile decisions, manual confirmation of steps is a necessity.

8. Business Benefits

With the development of the presented SLA Negotiation Framework in a business driven project like NextGRID, the research and design activities had a clear focus on usability in the business world. We therefore had to overcome problems and gaps related to the usage of SLAs in a business environment.

The identified requirements for a negotiation framework for e-business are:

- fast and effective advertisement of services to the outside world,
- fast and scalable search for matching service provider(s),
- building on technologies which are accepted as (de-facto) standards,
- flexible negotiation with adaptable pricing schemes, and
- cost-effectiveness.

With the existing design and implementation of the SLA Negotiation Framework we were able to address the first three requirements, whereas the basis for addressing numbers four and five was provided.

By using a service registry that can be queried by using a "Discovery Service" (in our case, we used UDAP), we enabled the fast and effective advertisement requirement. With the usage of global registries as well as the possibility of local registries, the yellow-pages concept was introduced which gives all service providers an equal chance to be discovered by the search mechanism (of course it depends on the offered service and quality levels).

In addition to the mechanism for publishing services, we concentrated also on the related search mechanism. The publication of services is only effectively usable (and will find its uptake in business), when services are discovered based on their capabilities (and the services' qualities). The introduced search delivers results that are of higher quality (in terms of fitting the requested services) the better the search parameters are defined. This allows for "global search" (imagine looking for a credit card validation service in the yellow pages), more detailed requests (a credit card validation service in the Netherlands), and fine granular searches (a credit card validation service in the Netherlands that is accredited with the financial administration).

In the overall evolution of the system, we always had an eye on the use of standard technologies and protocols where possible and sensible. This is an important aspect as the conformity to standards lays the basis for interoperability.

Early in the lifetime of the NextGRID project, the decision was taken to make use of a Discrete-Offer-Protocol for SLA negotiation. With that, we could not fulfil completely the requirement of flexible negotiations using different pricing models. In a real world business environment this "multi-phase" negotiation is important due to competition in the market. Flexible pricing mechanisms are needed to compete with other players in the market. Our framework is designed in a way which enables us to easily accommodate different negotiation protocols and pricing schemes.

Last but not least, the requirement of cost-effectiveness was (at least) partially addressed. With our SLA Discovery and Negotiation Framework we enable service providers and customers to conclude contracts fast and in an automated manner with decreased effort on both parties' sides. Outsourcing the functions of components like the negotiator to a negotiation broker (as described in [10]) decreases costs for the respective parties even further as they can use these kinds of services without having to invest large amounts of money in infrastructure and expertise.

9. Conclusions

In this paper we described a framework to discover and negotiate Service Level Agreements automatically, focussing primarily on the protocols and the components needed. The framework has been implemented as outlined and its ability to operate has been demonstrated to at various occasions.

The demonstrator shows that the negotiation and establishment of SLAs is a viable solution for electronic contracting. In addition to showing the viability, it also showed that many steps needed for establishing such a contract can be automated. We firmly believe that such automation is a crucial factor in the future adoption of SLAs in e-business environments and the proliferation of electronic service ecosystems. With the availability of frameworks and toolkits that facilitate the integration and use of SLAs, we expect the adoption of dynamic service sourcing based on SLAs to increase greatly and rapidly.

Another important aspect influencing the adoption speed of SLA-based e-business is the effort required to integrate SLA negotiation with existing or new applications. Although our demonstrator is specific to a particular scenario, our framework is implemented in a generic way that allows for the easy adaptation to different environments. Adding SLA negotiation capabilities to both customer and provider is therefore facilitated by our framework. It has to be noted, though, that actual enforcement of service levels is currently not handled by our framework.

With respect to the business validity of our approach we recognized early in the process of designing an SLA negotiation framework that the WS-Agreement protocol offers an instrument for a "supermarket-like" approach to SLA negotiation: the service provider offers a finite number of pre-defined services from which the customer can choose. Such an approach may serve a large number of application scenarios, but we also came across others which require a more sophisticated negotiation protocol. We therefore brought a group into being which comprises partners from a number of European projects to advance and standardise an SLA negotiation protocol. That protocol is aimed at the establishment of legally binding contracts. The work will also include research on the usefulness of different – in particular dynamic – pricing models. The results will be integrated into our framework once the new protocol has been specified.

Acknowledgments

This work has been supported by the NextGRID and BREIN projects and has been partly funded by the European Commission's IST activity of the 6th Framework Programme. This paper expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this paper.

References

[1] SLA Management Handbook - vol. 2 - Concepts and Principles, TeleManagement Forum, 2005.

[2] The NextGRID Project, Priority IST-2002-2.3.2.8. Web site, 30 April 2007 <http://www.nextgrid.org/>.

[3] S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, and I. Sedukhin, Web Services Resource 1.2 (WS-Resource), OASIS Standard, April 2006.

[4] B. Mitchell and P. McKee, SLAs A Key Commercial Tool, In P. Cunningham and M. Cunningham (eds.), *Exploiting the Knowledge Economy - Issues, Applications, Case Studies (Proc. of the eChallenges e-2006 Conference)*, Volume 3, IOS Press, October 2006.

[5] A. Andrieux et al, Web Services Agreement Specification (WS-Agreement), Grid Forum Document, GFD.107, Open Grid Forum, March, 2007.

[6] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems. *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2005.

[7] UNICORE, web site. 30 April 2007 <*http://www.unicore.eu/*>.

[8] T. Maguire, D. Snelling, T. Banks (eds.) Web Services Service Group 1.2 (WS-ServiceGroup), OASIS Standard, April 2006.

[9] eXist XML Database, web site. 30 April 2007 <*http://exist.sourceforge.net/*>.

[10] P. Hasselmeyer, C. Qu, L. Schubert, B. Koller, and Ph. Wieder, Towards Autonomous Brokered SLA Negotiation, In P. Cunningham and M. Cunningham (eds.), *Exploiting the Knowledge Economy - Issues, Applications, Case Studies (Proc. of the eChallenges e-2006 Conference)*, Volume 3, IOS Press, October 2006.