
Article:
Sharaf, S and Djemame, K (2014) Enabling service-level agreement renegotiation through extending WS-Agreement specification. Service Oriented Computing and Applications, 9 (2). 177 - 191. ISSN 1863-2386

https://doi.org/10.1007/s11761-014-0159-5

Service Oriented Computing and Applications manuscript No.
(will be inserted by the editor)

Enabling Service Level Agreement Renegotiation Through Extending WS-Agreement Specification

S. Sharaf · K. Djemame

Received: date / Accepted: date

Abstract WS-Agreement is a language and protocol designed for creating Service Level Agreements (SLAs) based on initial offers, and for monitoring those offers at runtime. The definition of WS-Agreement protocol is very general and does not contemplate the possibility of changing an agreement at runtime. This paper presents extensions of the WS-Agreement specification to support the dynamic nature of SLAs by allowing the possibility of SLA renegotiation at run time. The extended WS-Agreement specification have been implemented and tested. Within this implementation, the concept of renegotiation is demonstrated through the ability to create more than one SLA at runtime. An evaluation is conducted to examine the profits a service provider may gain through renegotiation, as well the savings resulting from rescuing the SLA from violations as a consequence of avoiding paying penalties. The results show that making the SLA terms adaptable and changeable is a viable mechanism that provides flexibility to the service provider and service consumer.

Keywords Dynamic Service Level Agreement (SLA) · WS-Agreement · Quality of Service · Renegotiation · Fuzzy systems.

Grant sponsor: FP7-ICT-2009-5 contract 257115 Optimized Infrastructure Services (OPTIMIS).

School of Computing
University of Leeds, Leeds LS2 9JT
Tel.: +44 113 3436590
Fax: +44 113 3435468
E-mail: ssharaf@kau.edu.sa, k.djemame@leeds.ac.uk

1 Introduction

Advances in Grid/Cloud computing research have in recent years resulted in considerable commercial interest in utilising infrastructures such distributed environments provide to support commercial applications and services [33]. The dependency on Grid/Cloud systems accelerated the need for replacing the best-effort approach used in these environments with a more controlled and reliable one to achieve the high levels of Quality of Service (QoS) necessary to potential users. In commercial Grids/Clouds, it is vital to define a QoS assurance for service consumers and service providers since it is associated with cost to be paid by the service consumer in case a job or service is successfully completed, as well as a penalty to be paid by the service provider in case of non-fulfillment. This QoS assurance is delivered in the form of electronic contracts between the service provider and service consumer called Service Level Agreement (SLA). It depict the provided service explicitly in terms of the requirements, guarantee terms, and the responsibilities of each party.

Several XML language-based specifications and tools are proposed in the literature to describe the agreement between the service provider and the service consumer, e.g. the WS-Agreement specification from the Grid Resource Allocation Agreement Protocol (GRAAP) group [24]. The WS-Agreement is a Web Service protocol used to create the SLA between agreement parties and consists of three parts: a schema for specifying an agreement, a schema for specifying an agreement template, and a number of port types and operations used to manage the agreement life cycle from creation to termination and monitor the agreement states in between. This specification has been criticised by researchers and developers due to the lack of a negotiation process be-

tween the two parties prior to committing and signing the agreement. Another limitation of WS-Agreement is its static nature; once the agreement is signed, it is unchangeable during service operation. In addition to this, the SLA monitoring system should stick to the terms in the SLA and prevent them being breached.

The state of the SLA may be an important reason for reducing the reliability and trustworthiness of parties if an unexpected event occurs at runtime, which may affect the state of the SLA. Therefore, it is not possible to adapt the terms or the (negotiated) QoS parameters of the agreement to accommodate this new state. The main motivation of this work is the move towards more flexibility and reliability in SLA support and fulfillment. The main challenge escalates from the dynamic nature of Grids/Clouds: the possible occurrence of any events at runtime may have either positive or negative consequences on the level of quality agreed by both parties - the service provider and the service consumer - through the SLA. In such systems, it is essential to maximise profit, minimise SLA violations and complete a maximum of tasks successfully. The aim of this research is then to make the SLA terms adaptable and changeable by allowing the agreement parties to renegotiate the guarantee terms at runtime while the service is in operation. To achieve this, the current WS-Agreement specification is limited and therefore needs to be extended to support dynamic SLAs through the possibility of renegotiation. A dynamic SLA in this research refers to the opportunity for agreement renegotiation while the service is being executed.

The contributions of this paper include:

1. introduction of dynamic SLAs by extending the WS-Agreement specification to enable SLA renegotiation at runtime. The extension affects several parts within the specification: the agreement structure, operations and states;
2. a decision support system to guide and support agreement parties - the service provider and the service consumer - whether to accept the new agreement generated after the renegotiation or stick to the initial one. This unbiased assessment of the renegotiation is based on Fuzzy Logic;
3. an implementation framework of the extended WS-Agreement specification is introduced to demonstrate the ability to create more than one SLA at runtime and initiate the renegotiation process by either party: the service provider or service consumer. The implemented framework has been evaluated to assess the performance of the new WS-Agreement extension and compare the outcomes generated by simulating different scenarios (optimistic and pessimistic) between a static SLA and the new proposed dynamic SLA.

This research has considered SLAs in Grid computing and the proposed contributions can equally be applied in a cloud environment.

The remainder of the paper is structured as follows: in Section 2, the motivation scenarios covered in this research are presented while Section 3 introduces some related work. Section 4 describes the proposed extensions of WS-Agreement to support re-negotiation at run-time and Section 5 presents their implementation. Section 6 discusses the results of the experiments that were designed to evaluate the benefits of dynamic SLAs. In conclusion, Section 7 provides a summary of the research.

2 Motivating Scenarios

Negotiation is a process between a service consumer and a service provider to reach an acceptable agreement offer from an initial agreement template. A typical example is the negotiation of a service provisioning time in co-allocation scenarios, or the negotiation of related service parameters such as the number of resources that are provided by a service and the price of the service.

At runtime, the service provider and the service consumer may face unexpected situations, which may force both parties to update their agreed pre-runtime QoS requirements. Thus, renegotiation of existing agreements applies the same signaling pattern as negotiation of agreements. A typical example is the renegotiation of an existing agreement in order to cope with peaks in a service usage.

These situations can be either *optimistic* or *pessimistic*. An optimistic scenario can be viewed as a series of events taking place at runtime which may trigger an improvement in the level of QoS agreed pre-runtime. For example, this enhancement in QoS can be through reducing the service probability of failure (PoF), completing the service execution sooner, or any other beneficial situation to either party. On the other hand, a pessimistic scenario is viewed as the occurrence of events which may force either party to decrease the QoS level agreed pre-runtime. This reduction in QoS level can be a potential solution to rescue the running service from failure and consequently prevent the SLA violation. The initiation of the renegotiation procedure can be from either the service provider or the service consumer perspective. Four different scenarios are considered in this research:

– **Provider-Optimistic (PO)**: From the service provider perspective, an optimistic scenario correlates with

increased profit and better resource management. A typical example is while the service is running extra resources become available in the provider's infrastructure, following the completion of another service. Consequently, the service provider may decide to provide these extra resources to the negotiated service to speed it up and gain extra profit. This is achievable through the renegotiation at runtime with the service consumer by making an offer (*service completion time shortened, higher price*). If the latter is able to pay more, then both parties will benefit from this renegotiation, especially the service provider.

- **Provider-Pessimistic (PP)**: Once an SLA has been agreed and resources have been allocated to the negotiated service, some of these resources may fail. This can cause a pessimistic scenario for the service provider. Although the service provider is expected to have some fault tolerance mechanisms in place, there is no guarantee that the provider would be able to complete the service execution successfully, or on time. If SLA renegotiation, triggered by the service provider, is possible then the service provider can make an offer to the service consumer (*service completion time extended, lower price*) to prevent SLA violation.
- **Consumer-Optimistic (CO)**: An optimistic scenario may also involve a service consumer getting a service with higher levels of quality than the one agreed pre-runtime. In this case, SLA renegotiation is triggered by the service consumer and an offer is made to the service provider, for example to speed up the service execution thanks to the allocation of extra resources at higher cost (*service completion time shortened, higher price*). Both parties are expected to benefit from this renegotiation, especially the service consumer.
- **Consumer-Pessimistic (CP)**: A pessimistic scenario may also involve a service consumer getting a service with lower levels of quality than the one agreed pre-runtime, for example following a drop in the consumer's actual budget. In this case, SLA renegotiation is triggered by the service consumer and an offer is made to the service provider, for example to extend the service execution thanks to the allocation of different quality resources at lower cost (*service completion time extended, lower price*).

3 Related Work

There are various approaches in the field of SLA management in distributed environments. The related area of SLA negotiation is a popular research topic within the wide Grid community, which can be seen in some EU funded projects such as Brein [4], NextGrid [5], BEinGRID [6], smartLM [7], SLA@SOI [27] and RESERVOIR [8]. These have all promoted the use of SLAs and developed basic Grid components and architectural support for negotiating SLAs to aid in the consumption of services between service consumer and service provider. It is worth mentioning the CONTRACT project which produced a new language for the expression of contracts between Web services [34]. Unlike SLAs which are based in a set of computer-observable parameters, this language takes into account the computational issues of reasoning over contracts and, specifically, verifying the properties of systems determined by sets of contracts. The IRMOS project [12] proposes an SLA framework which introduces a chain of linked SLAs implemented on different layers in order to provide support for the provision of real-time applications.

The Grid Resource Allocation Agreement Protocol Working Group (GRAAP WG) of the Open Grid Forum (OGF) has produced the Web Services Agreement (WS-Agreement) standard [24] to create bilateral agreements. In many cases, SLAs are modelled following both service consumers and service providers' business objectives [25], ensuring their management provides QoS guarantees at the same time.

Examples to date of WS-Agreement implementations include WSAG4J [3], Cremona [22], AssessGrid project [28,32], and the SORMA project [23]. WSAG4J considers only direct negotiation between service consumer and service provider. It is an actively developed implementation of WS-Agreement and publicly available. Cremona was developed by IBM using an early version of WS-Agreement but only considered direct negotiation between consumer and provider. The project remains inactive with closed source code and no status updates.

Ludwig et al. [17] describe the use of WS-Agreement for SLAs paving the way for using multiple distributed resources to satisfy a single service request. Frankova et al. [16] provide a formal definition of an agreement and analyzing the possible evolutions of agreements and their terms over an execution. Therefore, they identify a number of extensions which involve the initial negotiation, the monitoring of running agreements, and the possibility of renegotiating agreements in executions. Pichot et al. [15] propose and discuss extensions to the WS-Agreement protocol which support dynamic negotiation and creation of SLAs in an efficient and flexible manner. Wieder et al. [18] give an overview of state-of-the-art Grid software using SLAs in the domain of scheduling and resource management. Waeldrich et al. [29] present similar work which led to the WS-Agreement

specification. They describe a Web Services protocol for negotiating agreement offers between the two parties, service consumer and service provider. Battre et al. [28] describe the Web Services Agreement Negotiation protocol proposed by the Open Grid Forum to extend the existing specification. This proposal is the result of combining various research activities that have been conducted to define protocols for negotiating service levels or to supersede the existing "take-it-or-leave-it" protocol. The main characteristics of this proposal are the multi-round negotiation capability, renegotiation capability, and compliance with the original specification. Kuebert et al. [14] present an approach for the implementation of an SLA framework which allows negotiation and renegotiation of QoS requirements. The framework is specifically designed to guarantee the correct deployment and execution of soft real-time interactive applications over Service Oriented Infrastructures (SOIs), respecting the interests from all involved parties. Menychtas et al. [13] present a novel cloud platform, which was developed in the frame of the IRMOS project targeting soft real-time applications that have stringent timing and performance requirements. Their platform combines SOIs with virtualisation technologies to manage and provision computational, storage and networking resources as well as to communicate with legacy systems such as WiFi locators.

Renegotiation has not been seen as a necessity but is desirable in case of the need of more resources, the prolonging of the agreement or the release of already reserved resources. Parkin et al. [19] described an abstract, domain-independent SLA protocol for the renegotiation of contracts. It is based on the principles of contract law to make agreements with it legally-compliant and allows for multi-round renegotiation in an environment where messages may be lost, delayed and reordered. Di Modica et al. [21] proposed an extension of the WS-Agreement which allows both parties to renegotiate agreements.

The research presented in this paper discusses runtime re-negotiation of SLAs between two parties - service consumer and service provider - using an extension of the WS-Agreement specification agreement factory construct. A framework implementation of the extended WS-Agreement specification is introduced to demonstrate the ability to support dynamic re-negotiated SLAs at runtime. Close work to this research includes [29,28,20,21,16,19,15]. However, as explained in the evaluation section (se section 6.4) this research differs in many aspects, which include WS-Agreement protocol extension, WS-Agreement protocol implementation, schema extension, renegotiation at runtime, consideration of optimistic and pessimistic scenarios, and decision support provision in SLA renegotiation.

4 WS-Agreement Specification Extension

Th extended WS-Agreement protocol must specify the required interfaces to renegotiate existing agreements. In this context, re-negotiation of agreements is considered to be a bilateral process, which results in a re-negotiated agreement. Therefore, the capabilities to create re-negotiated agreements based on initial negotiatied offers need to be defined.

This research assumes a direct communication between the service consumer and the service provider. Both parties must agree pre-runtime what terms are renegotiable at run time. This takes place when the service provider provides a negotiation template where fixed terms and renegotiable terms are clearly defined. At runtime, either party (service consumer and service provider) can initiate SLA renegotiation according to specific conditions (see section 2), which may lead to the agreement of a new SLA. One of the objectives in this work is to uncover the behaviour of both parties at runtime through analysing the changes required to the SLA during the service execution. This vision of dynamic SLAs renegotiable at run time cannot be supported with the current WS-Agreement specification. The following sections describe the different extensions made to the specification.

4.1 Agreement Structure

Once an SLA is agreed pre run-time between the service consumer and service provider, and with the possibility of renegotiation, it is important to keep track of the negotiated/re-negotiated agreements through the service life cycle. For this reason, a new variable of type Endpoint Reference (EPR) is added in the context section of the agreement to contain a reference to the next renegotiated agreement. Its default value in creating an agreement will be (null) and after a successful renegotiation process, the value of this variable will hold the EPR of the newly created agreement. The Service Level Objective (SLO) data type in the Guarantee term specifies the level of service that must be met to fulfill the Guarantee term. As mentioned previously, the service provider must specify which SLOs are static and which are dynamic during renegotiation. For this reason, extra information is added to the SLO to indicate whether this SLO can be modified during the agreement renegotiation and therefore any QoS aspects can be consid-

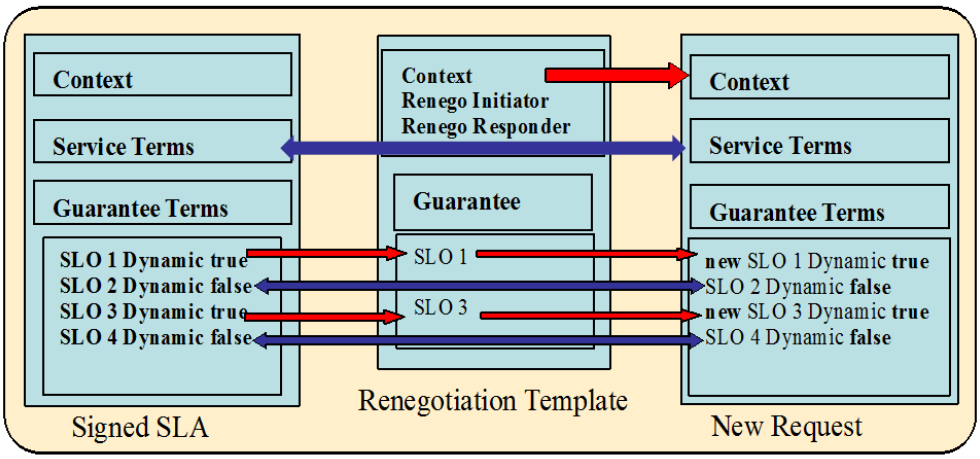


Fig. 1: New Request Creation From Renegotiation Template

ered in the renegotiation by defining their SLO to be dynamic.

4.2 Renegotiation Template

During the renegotiation process, both parties will only re-negotiate a number of SLOs, not the entire agreement. A new template called *RenegotiationTemplate* is introduced to support the dynamic SLOs update. This template consists of a context section that has information on the renegotiation initiator and the renegotiation responder, and another section on the Guarantee Terms holding the dynamic SLOs only. Figure 1 shows how to build the new request from the renegotiation template during the renegotiation process using the SLA agreed pre-runtime.

4.3 Agreement State Machine

In the WS-Agreement specification, the agreement has runtime states describing the state of the agreement. The defined set of agreement states within the current WS-Agreement is *Pending, Pending and Terminating, Observed, Observed and Terminating, Rejected, Completed, Terminated*. Enabling SLA renegotiation at runtime necessitates an extension of the possible states in the agreement lifecycle.

There is a wide number of renegotiation scenarios, depending on whether a service consumer or a service provider initiates the negotiation process, which party creates the negotiated agreement, and where the resulting agreement state is hosted. In the WS-Agreement extension with the possibility of renegotiation the agreement states must monitor the agreement and initiate the renegotiation protocol. To accommodate this, new states as illustrated in Figure 2 are added to the WS-Agreement specification: *Checking, Warned, Violated, Hold*:

- **Checking:** When the service execution starts, the SLA is monitored and enters the new state **Checking**. This state follows **Observed**.
- **Complete:** This state means that the agreement has been executed and finished successfully. This state follows **Checking**.
- **Warned:** In a pessimistic scenario, potential events may lead to SLA violation. As a result, the SLA state is moved to **Warned**. This state follows **Checking**.
- **Hold:** This state means that either party wish to renegotiate the SLA. For example, the service provider is able to free up some extra resources and provide a better service to the consumer, or the ser-

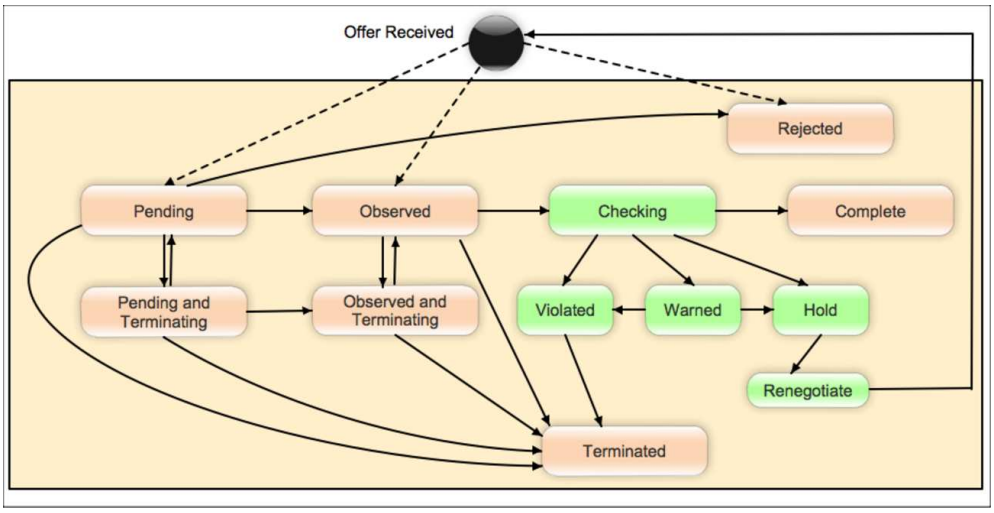


Fig. 2: Extended Agreement States

vice consumer is willing to pay a higher price than previously negotiated in order to get more reliable resources. Consequently, the SLA state moves to **Hold**. This state follows **Checking** or **Warned**.

– **Violated:** While monitoring the agreement, there is a possibility of breeches occurring in SLA terms so the agreement will transit to the **Violated** state. This state follows **Checking** or **Warned**.

4.4 Renegotiation Protocol

The agreement renegotiation can be initiated by either party: the service provider or the service consumer. Hence, the next section illustrates two different protocols according to the renegotiation initiator/responder role.

4.4.1 Initiated by Service Consumer

The process illustrated in Figure 3 represents the renegotiation process at runtime when initiated by the service consumer. Renegotiation takes place according to the following steps:

– The service consumer requests the renegotiation template from the service provider side. It is worth noting the possibility for the service provider to make the re-negotiation template available to the service consumer once the negotiation (pre-runtime) is completed and the SLA is signed. Thus, a local copy of the renegotiation template is available to the service consumer.
– Once this template is received, the service consumer makes the necessary amendments to the SLOs according to the new QoS level required. The new SLA request builds on the initial SLA agreed pre runtime and the dynamic SLOs as shown in Figure 1 .
– Once this request is ready, the service consumer sends it to the service provider to get a new quote.

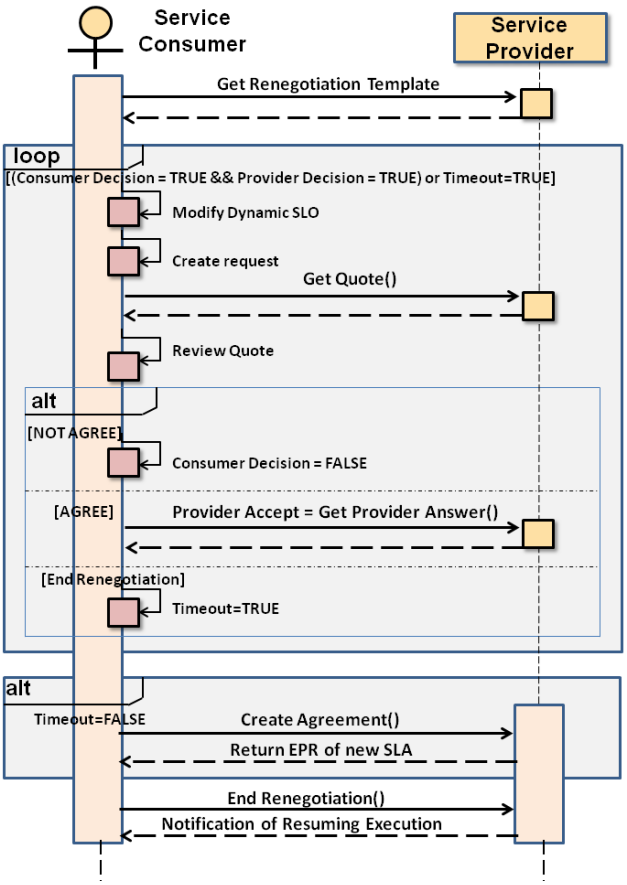


Fig. 3: Renegotiation Initiated by the Service Consumer

– On the provider side, a new quote is then provided according to the request and sent back to the consumer. Note that the service provider may rely on recommendations (Accept/Reject) of a decision support system, e.g. a fuzzy system, see section 5.3.
– The consumer reviews the quote to make a decision (Accept/Reject). This process is repeated until both parties agree the new SLA after renegotiation or the renegotiation allowable time is over.
– If the new offer is accepted then a new SLA is signed and the execution resumes. Otherwise, the execution resumes and the original SLA still holds as a new agreement has not been reached.

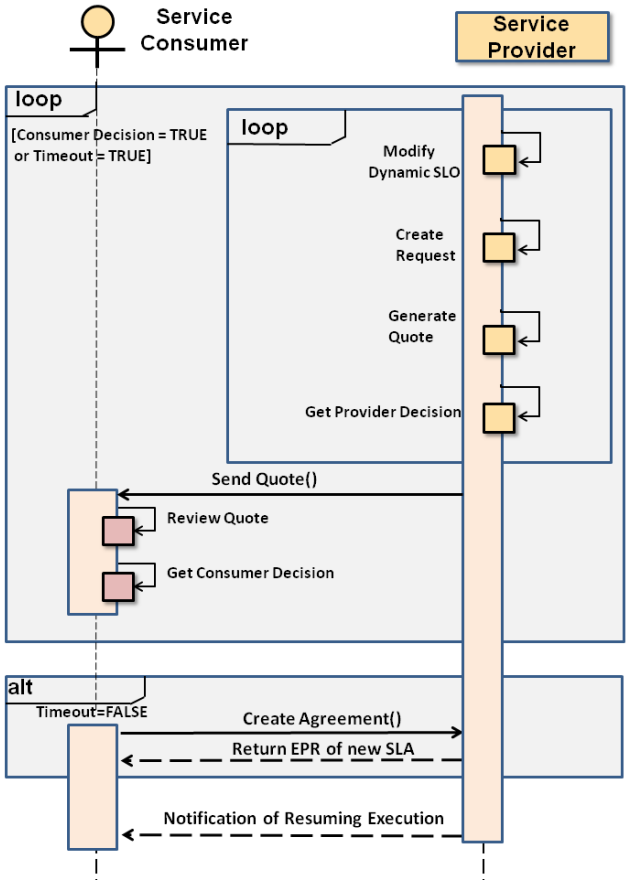


Fig. 4: Renegotiation Initiated by the Service Provider

4.4.2 Initiated by Service Provider

At runtime, the service provider may be able to provide a better service (optimistic scenario) or unable to fulfill the SLA (pessimistic scenario). Consequently, a renegotiation process with the service consumer is initiated as illustrated in Figure 4.

– The service provider starts modifying the NegotiationTemplate by amending the SLOs according to the new QoS level, building up the renegotiation request, and getting new quote(s). Note that the service provider may rely on recommendations of a decision support system, see section 5.3.
– Once a new quote satisfies the service provider, a notification message alongside a new quote is sent to the service consumer about initiating the renegotiation process.
– This process is repeated until both parties agree the new SLA after renegotiation or the renegotiation allowable time is over.
– If the service consumer accepts the new offer, then a new SLA is signed and the execution will resume. Otherwise, the execution resumes using the original SLA.

5 Extended WS-Agreement Implementation

In order to implement WS-Agreement extensions, the question arises regarding which existing WS-Agreement implementation framework to build upon, especially to support the renegotiation process (see section 3). Among the existing WS-Agreement implementations, the AssessGrid project WS-Agreement implementation [28] was chosen due to the authors' involvement in its usage and evaluation [32].

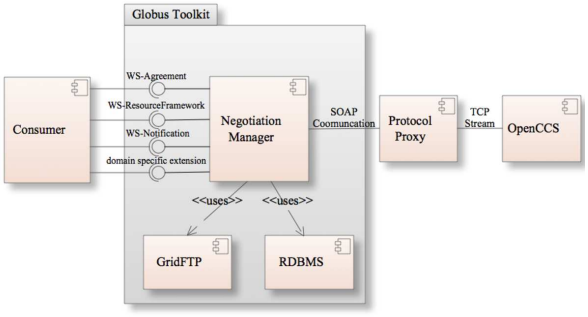


Fig. 5: Negotiation Manager

5.1 Negotiation Manager

This WS-Agreement implementation has the *Negotiation Manager (NegMgr)* as its main component [28], which is responsible for negotiating SLAs with system-external contractors. Consequently, it needs to provide an interface to the surrounding world to be addressable by possible contractors. Well-used communication

protocols ease the communication with the contractors. The Negotiation Manager is implemented as a Globus Toolkit 4.0 service, which allows to use and support several mechanisms that are offered by the toolkit, such as authentication, authorization, GridFTP, and monitoring services [35]. Figure 5 shows an example of how it interacts with the service consumer as well as OpenCCS, a planning based and topology aware Resource Management System (RMS) which computes a complete job schedule about future resource usage and assigns a start time to all jobs, as described in [28,35].

5.2 WS-Agreement Extensions

WS-Agreement extensions implementation is explained next, with an illustration of the relevant operations that are required in the renegotiation.

GenerateTemplate. Figure 6 shows the flow in the operation *GenerateTemplate*, which is called after a party initiates the renegotiation process by sending a request containing the current SLA with the identity of the renegotiation initiator. Generating the renegotiation template consists of two stages: 1) building the renegotiation context which includes the parties and their roles, and 2) building up the terms section and including the dynamic terms.

Renegotiate. This operation is responsible for modifying the dynamic SLOs, which can be changed during renegotiation, e.g. a service execution time. The operation consists of loading the guarantee terms in the renegotiation template as well as the SLOs for modification.

Generate Quote. Following the modification of the dynamic SLOs and the request for a new SLA quote, the service provider invokes the operation `Generate Quote`, which returns a list of quotes. The provider has general constraints for executable services according to its individual provided resources (hardware and software). During renegotiation the interaction with the RMS takes place through the Negotation Manager, as the Negotiation Manager is the link between the RMS and the actual infrastructure (Grid/Cloud). An Offer Manager (OM) is responsible for SLA quote and agreement creation depending upon not only which scenario an SLA quote or agreement request is made but the service provider policies as well [28].

Provider Decision. The service provider can assess the renegotiation *benefit* thanks to the use of a decision support system (see Section 5.3). It also runs a final compliance test with the renegotiated SLA template to ensure that the new SLA offer(s) is/are correctly formatted.

Review Quote. Use of this method denotes that the service consumer receives a list of SLA quotes ready to review. To rank these offers, the service consumer may use Dempster-Shafer Analytical Hierarchy Process (DS-AHP) [38,32]. The service consumer can also assess the renegotiation *benefit* thanks to the use of a decision support system by invoking **Consumer Decision** operation (see Section 5.3).

5.3 Fuzzy Support System

After a successful renegotiation a new agreement is formed. However, before committing to such agreement, either party (service consumer and service provider) should assess its *benefit* since both parties may have conflicting objectives. To support this assessment, a fuzzy logic decision support system is designed to analyse the difference between the initial agreement (pre-runtime) and the renegotiated one. This analysis is based on evaluating the changes introduced in the renegotiated agreement such as price, and provides a number of outputs so that both parties can decide whether to accept the new SLA or reject it. Usage of such decision support system is described in section 6.2.

6 Evaluation

6.1 Case Study

To illustrate the implementation of the WS-Agreement extensions and the usage of the fuzzy logic decision support system, a case study in risk management in Grid computing is considered [31,32] and involves an end-user and a service provider.

An end-user (service consumer) is a participant from a broad public approaching the Grid in order to perform a task comprising of one or more services. The user must indicate the task and associated requirements formally within an SLA template. Based on this information, the end-user wishes to negotiate access with providers offering these services, in order that the task is completed. The end-user must make informed, risk-aware decisions on the SLA quotes it receives so that the decision is acceptable and balances cost, time and risk.

A service provider offers access to resources and services through formal SLAs specifying risk, price and penalty. Providers need well-balanced infrastructures, so they can maximise QoS and minimise the number of SLA violations. Such an approach increases the economic benefit and motivation of end-users to outsource their IT tasks. A prerequisite to this is a provider's trustworthiness and their ability to successfully deliver

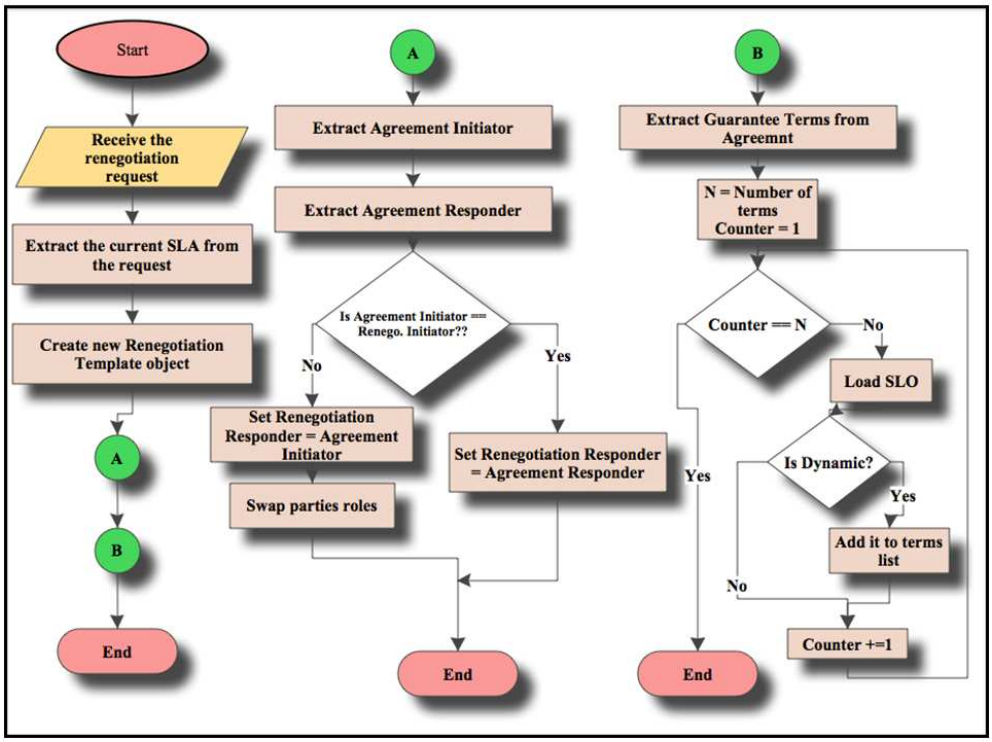


Fig. 6: Operation GenerateTemplate Flow Chart

an agreed SLA. Assessments of risk allow the provider to selectively choose which SLA requests to accept.

The use of a brokering service can also be envisaged. A broker acts as a matchmaker between end-users and providers, providing a risk optimised assignment of SLA requests to SLA quotes. It is responsible for matching SLA requests to resources and services, which may be operated by an arbitrary number of providers. The broker's goal is to drive this matchmaking process to a conclusion, when the provider will propose an SLA. More details are available in [31,32].

6.2 The Fuzzy System

As explained in Section 5.3, a fuzzy logic decision support system is designed to analyse the difference between the initial agreement (pre-runtime) and the renegotiated one, and assess the benefit of renegotiation. To build the fuzzy system, the definition of the linguistic variables and their classification as inputs and outputs is required for the case study. The three main parameters describing an SLA are:

1. The risk (represented as a Probability of Failure (PoF)) of the SLA failure.
2. The price to be paid by the service consumer.
3. The penalty fee to be paid by the service provider should the SLA fail.

Let denote PoF_i, $Price_i$ and $Penalty_i$ the PoF, price and penalty in the SLA initially negotiated pre-runtime respectively, and PoF_r, $Price_r$ and $Penalty_r$ the PoF, price and penalty in the renegotiated SLA. The fuzzy system is based on the differences (parameter value variation) between the pairs $\langle PoF_i, PoF_r \rangle$, $\langle Price_i, Price_r \rangle$, and $\langle Penalty_i, Penalty_r \rangle$, and are considered as inputs in the fuzzy sytem. The fuzzy system outputs, which value fall in the interval [0..1] are:

1. Benefit of Accepting (BoA): this is the benefit of accepting the new SLA after renegotiation for the service execution only, without prioritising either party's benefit.
2. Benefit for Provider (BfP): this is the benefit for the service provider, i.e. how much the new SLA after renegotiation is beneficial for the service provider.
3. Benefit for Consumer (BfC): Since the consumer may not has the monitoring tools and other statestical services as a renegotiation guidance and to assisst him in accepting/rejecting the new agreement. This parameter will represent the benefit for consumer the same as BfP.

Each input can be described by a linguistic value (negative, zero, positive) whereas each output can be defined as (low, medium, high). The definition of the fuzzy rules will interpret the relationships between the inputs and outputs. The fuzzy rules are constructed

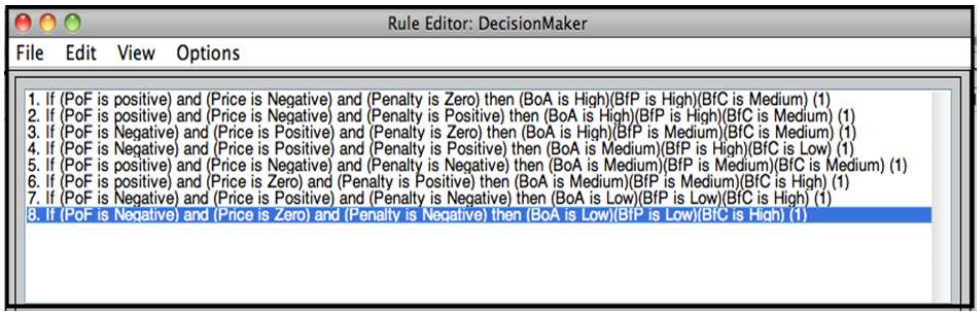


Fig. 7: Fuzzy System rules

in the logical form: `(IF x is A THEN y is B)` where A, B are linguistic values and x, y are linguistic variables, e.g. `(IF PoF is Negative then BoA is Low)`. The combinations of the input linguistic values describe the different scenarios which may take place. In the following these combinations are limited to 8 possible cases as shown in Figure 7. These rules control the relation between variances in the inputs and result in the expected output linguistic value for each variable.

6.3 Results

The service deployed is tested using a task SLA request sent initiated by a service consumer. The node has a quad core Intel Core 2 CPU running at 2.4Ghz with 4Gb RAM and 500GB storage. The running environment of the negotiation manager is described as follows: Linux operating system with the Debian 64-bit distribution and Globus Toolkit version 4.0.8. Besides Globus, additional applications include Java version 1.6.0.12 SE runtime Environment.

6.3.1 Fuzzy System Evaluation

The fuzzy inference system is built using Matlab fuzzy logic toolbox [37]. Once the linguistic variables together with the fuzzy member functions and fuzzy rules are defined, a number of plots describing the mapping of inputs to outputs according to the member functions and rules are generated.

In this first experiment, the price and penalty associated with the SLA are set with their corresponding linguistic values in the range [-1000..1000].

Figure 8 shows the relationship between PoF, price and the resulting BoA, BfC, and BfP. PoF and price are usually inversely proportional which means that a PoF decrease results in a price increase with the provision of, e.g. better service quality, and vice versa. Four areas are labelled A, B, C and D on the plot. Area B contains the lowest value of BoA due to the PoF increase and price increase. BoA has a mid-value of 0.5 as shown in area C, where a decrease in PoF and price is shown. In area D, a PoF decrease and price increase lead to the highest BoA, as shown in area D. In Area A has a slightly lower value of BoA is shown due to a PoF increase and consequently a price decrease.

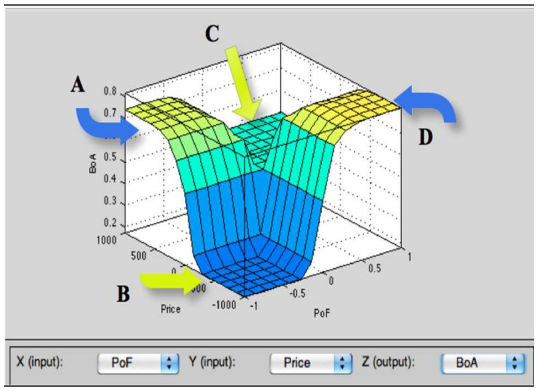
For BfC, area B shows its maximum value whereas for BfP it shows it lowest wit a PoF increase and a price decrease. Area C has an improved value for BfC compared to BoA due to a decrease in both PoF and price. This is seen as clear benefit for the service consumer, although one that may rarely occur.

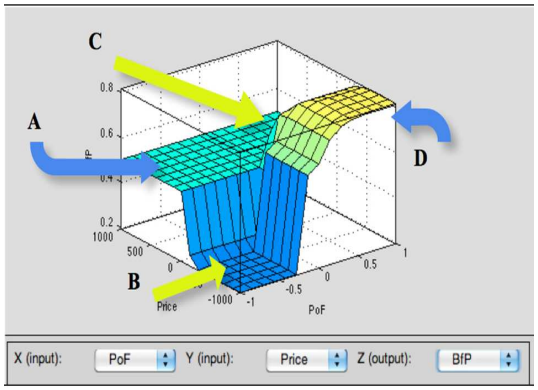
For BfP, from the service provider perspective, having a renegotiated SLA with an increased price and decreased PoF is a beneficial situation which suggests this new SLA should be accepted, as illustrated in area D. Areas A and C show identical results for the service provider when the price decreases, with a BfP value of 0.5.

Similarly, Figure 9 shows the relationship between PoF, penalty and the resulting BoA, BfC, and BfP. BoA is determined according to the changes in PoF and penalty. The highest BoA value is just above (0.6) occurs between areas A and B, A and C, C and D. When there is no change in the PoF value in the renegotiated SLA and the penalty is decreased, it is recommended to accept the new SLA. Area B shows an increased PoF with an increased penalty, which is a pessimistic scenario leading to low BoA.

Next, the highest value of BfP is 0.8 in area A. This is due to a PoF increase and penalty decrease (which is not common but may occur with a price decrease as well). This is viewed by the provider as a good opportunity to accept the renegotiated SLA.

Finally, BiC proves again that both parties have opposite objectives: area A was described earlier as a good opportunity for the provider but not for the consumer in this case. However, area B shows a good opportunity for the consumer to get an increased penalty with a PoF increase.





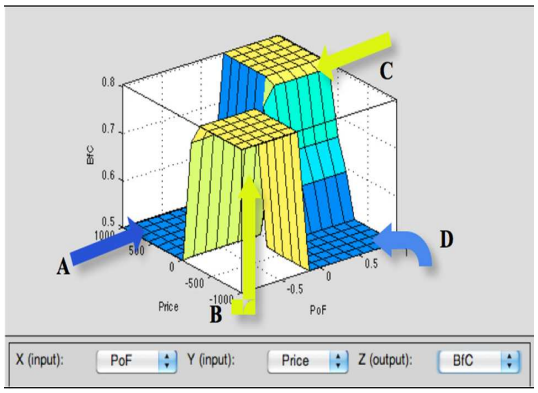
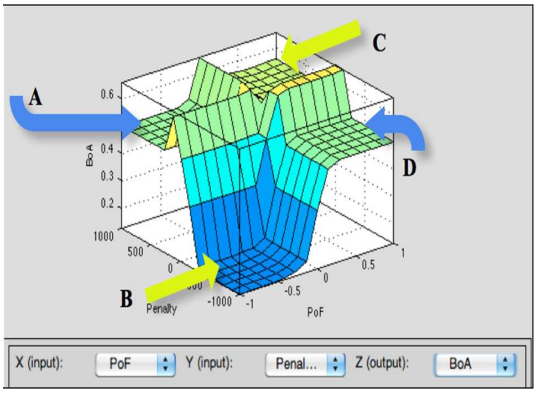
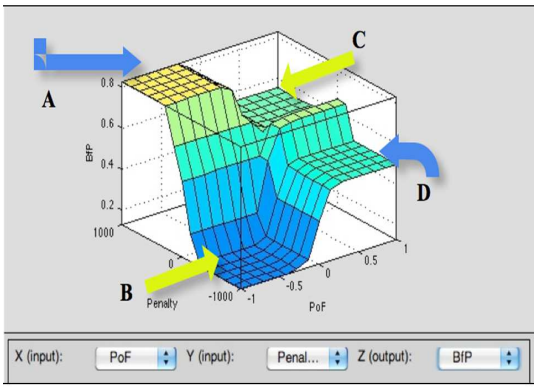


Fig. 8: PoF, Price, and the resulting BoA, BfC, and BfP





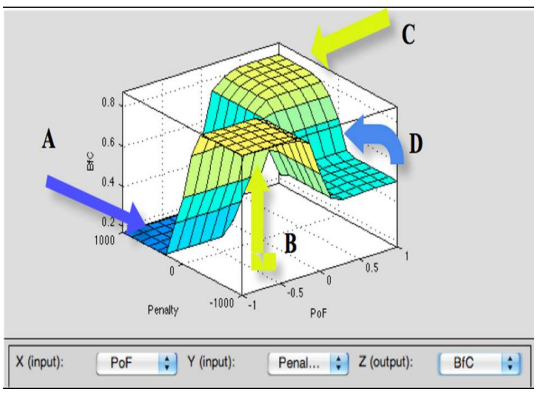


Fig. 9: PoF, Penalty, and the resulting BoA, BfC, and BfP

6.3.2 Renegotiation - 4 SLAs

Evaluating the extended WS-Agreement is made through using the implemented framework to ensure the possibility of SLA renegotiation at runtime and creation of new agreements. Further evaluation focusses on the analysis and assessment of the benefit returned to both parties following renegotiation.

The PoF, price, and penalty parameters are initially set to 0.15, 700, and 1000 respectively once the SLA is negotiated and agreed pre-runtime. At runtime, the SLA is renegotiated considering four scenarios in the following order: 1) provider pessimistic: 15% PoF increase, unchanged price, and 10% penalty increase; 2) provider optimistic: 10% PoF decrease, 20% price increase, and 10% penalty increase; 3) consumer opti-

mistic: 10% PoF decrease, unchanged price, and 10% penalty decrease, and 4) consumer pessimistic: 15% PoF increase, 10% price decrease, and 10% penalty decrease.

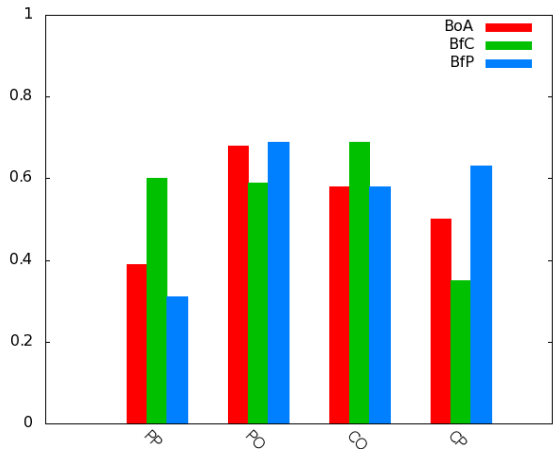


Fig. 10: Renegotiated SLAs and the resulting BoA, BfC, and BfP under various scenarios.

BoA, BfC, and BfP resulting from the renegotiated SLA are shown in Figure 10.

6.3.3 Renegotiation - 100 SLAs

In the third experiment, 100 events were generated randomly to simulate the four scenarios, and new values set in the SLA (PoF, price, penalty) after renegotiation are calculated according to the scenario under consideration. Figure 11 shows a) the different values for PoF, and b) price and penalty in 100 SLAs with a price increase of up to 300% (from 300 to 1215) and a penalty decrease of up to 74% from 1000 to 260. Although the PoF has a slight increase in general by 6% from (0.3 to 0.32), it reached a peak value of 0.84 during execution.

In order to compare static and dynamic SLAs, consider a high PoF, e.g. of 0.8. Assuming such probability is reached it is likely the SLA will be violated leading to the service execution ending with a failure, as shown in Figure 12. From the business perspective, the service provider will pay 1000 penalty to the service consumer in case of a static SLA, whereas in case the SLA is dynamic the service provider will still make 910 profit. In an optimistic scenario, a decrease in PoF in the renegotiated SLA means for the service consumer a higher QoS than in the original SLA and a price increase. This leads to a higher profit and a decrease in penalty for the service provider. On the other hand, in a pessimistic scenario, a dynamic SLA gives opportunities to prevent SLA violation and therefore penalty fees payment avoidance.

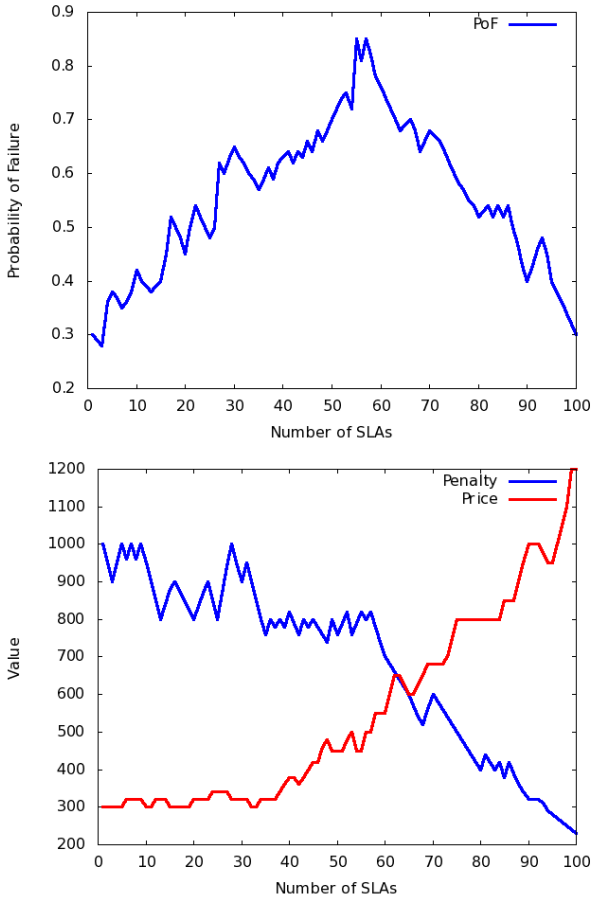


Fig. 11: Renegotiated SLAs and the resulting PoF, Price, and Penalty

Part of this experiment was to perform an evaluation of the overhead introduced by the renegotiation protocol. In this case, 100 SLA requests were generated to measure the renegotiation time, considering an optimistic scenario where renegotiation is instantiated by the service consumer requesting extra resources from the service provider. Figure 13 shows: 1) the time it takes the service consumer to construct a new SLA request, send it to the service provider, and for the provider to return a new offer; 2) the time it takes to receive an offer from the provider and agree to it, and 3) the total time taken for the renegotiation. Although application dependant, the time for scheduling extra resources through renegotiation in this scenario is acceptable (less than 2 seconds) in terms of QoS.

6.4 Capability Comparison

This section shows a comparison of this research with other work found in the literature considering the following criteria:

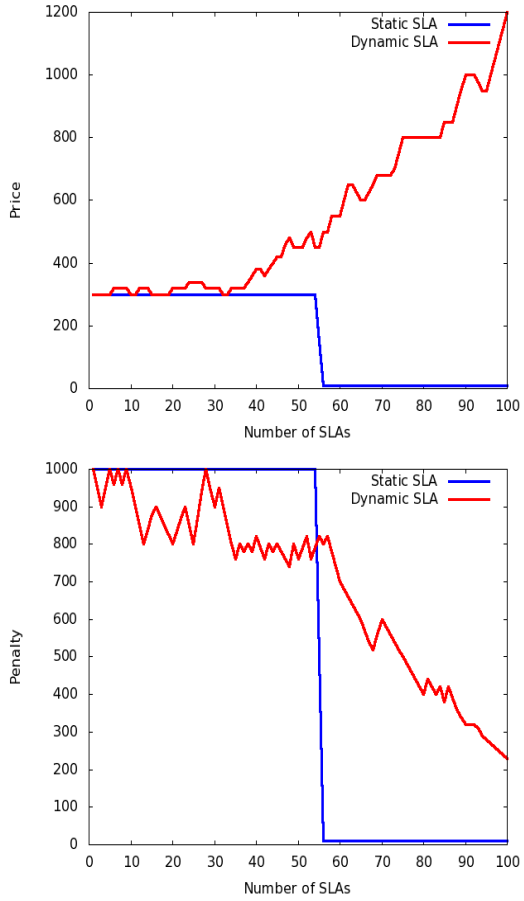


Fig. 12: Static and Dynamic SLA Comparison

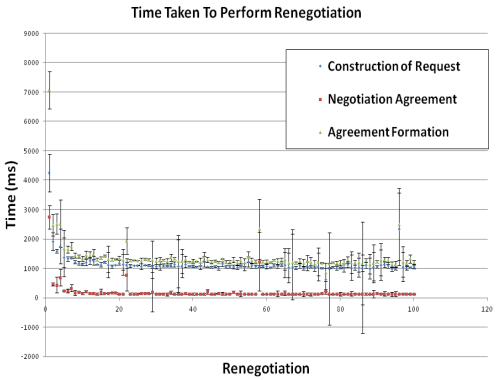


Fig. 13: Renegotiation Time

* **Extended Protocol** checks whether the protocol in the WS-Agreement has been extended or not.
* **Extended Schema** deals with the possible modifications made to the schema such as the agreement structure, adding new data types and new operations.
* **Renegotiation at runtime** examines whether renegotiation takes place at runtime when the service is in operation. Some research aims for the renegotiation to take place in the period between signing the initial SLA and the service execution start.
* **Scenarios** indicates possible events that may initiate the renegotiation procedure, e.g. optimistic or pessimistic. This criterion checks whether the events considered in the research affect the renegotiation outcome.
* **Decision Support** indicates the provision of value-added information to support both parties in deciding whether to accept the new SLA at renegotiation or not.
* **Implementation Availability** refers to whether this extended WS-Agreement has been implemented or not.

Table 1: Capability Comparison with related work

	[29] [28]	[20] [21]	[16]	[19]	[15]	This research
Extended Protocol	Yes	Yes	N/A	Yes	Yes	Yes
Extended Schema	Yes	Yes	Yes	No	No	Yes
Renegotiating At Runtime	Yes	Yes	No	N/A	No	Yes
Possible Scenarios	No	No	No	N/A	N/A	Yes
Decision System	No	No	No	No	No	Yes
Implementation Availability	N/A	N/A	N/A	N/A	N/A	Yes

The results of the capability comparison are shown in Table 1. This research tackles the problem of WS-Agreement renegotiation at runtime initiated by either party (service consumer or service provider) considering optimistic and pessimistic scenarios and supports all six comparison criteria.

The closest work to this research is by the GRAAP WG to produce an extension to the WS-Agreement called WS-Agreement Negotiation considering the introduction of negotiation/renegotiation when creating agreements [29]. In this extension, the negotiation is done as a separate process by adding an additional layer called Negotiation Layer on top of the Agreement Layer and Service Layer which are defined in the WS-Agreement Specification [24]. The renegotiation aspects are left for future work. However, in this research, the negotiation/renegotiation extension has been included in the Agreement Layer without adding an extra layer for ease and simplicity. This way the modification/tuning (createAgreement) operation works for both newly created agreements at negotiation pre-runtime, and renegotiated agreements at runtime.

6.5 Extensions

There are many ways to further refine and extend the work presented in this research. The most appealing ones are listed below:

- Introduction of a brokerage system where the renegotiation protocol handles the interaction between the service consumer and a broker, as well as the broker and the service provider.
- The consideration of other QoS aspects: the paper has shown how QoS dimensions such as service completion time and cost can drive SLA requirements. QoS aspects such as security and reliability can also be considered. In WS-Agreeemnt, service description terms give a functional description of the service to provide. Since the WS-Agreement is designed to be domain independent, the content of a service description term can be any valid XML document.
- The paper has considered the random generation of events to simulate service renegotiation at runtime. It will be interesting to deploy the extended WS-Agreement implementation in a real Grid/Cloud infrastructure. Some cloud computing-related projects have included the use of SLAs as an important component within their architecture, e.g. [36, 26]. Future work can involve analysing the current use of SLAs in Cloud Computing as well as the renegotiation aspects.
- The investigation of the renegotiation aspect in a Grid/Cloud infrastructure may involve the consideration of potential ripple effects of adopting new renegotiated SLAs at runtime. This is an important issue to study and handle for the service provider not only for infrastructures simultaneously running thousands of services, but also for composed services as well: amending the SLOs according to a new QoS level in a particular SLA may cause violations of other SLAs.
- An extended fuzzy system: the system evaluation presented in this paper shows compatibility between the Price and the Benefit-for-Provider (BfP) as well as between the Penalty and the Benefit-for-Consumer (BfC). A mathematical model can be used to define a pricing policy and penalty setting during renegotiation. These can help both parties to maintain their benefits through predefined values of BfP and BfC.
- The proposed approach in the paper suggests putting the service execution on hold during renegotiation. Halting the service execution during the renegotiation is viewed as a safe option, as not doing so may lead to an SLA violation. For example, a renegotiation requested by the service provider in case of a *Warned* state follows the possibility of not fulfilling a service completion time (pessimistic scenario), a situation the service provider would prefer to avoid. Further investigation is needed to assess whether renegotiation should take place while the service is executing, and therefore the results of the renegotiation should only affect the resources/priority assigned to the service invocation.

7 Conclusion

This paper presents an extension of WS-Agreement Specification for enabling SLA renegotiation at run-time.

A critical analysis of WS-Agreement limitations is provided. The definition of a dynamic SLA in the context of this research is given and the importance of SLA renegotiation is introduced through an illustration of various motivation scenarios. WS-Agreement extensions are proposed starting with an overall vision of the SLA renegotiation procedure at runtime. To reach this vision, several extensions in different parts of the WS-Agreement specification are added and include the agreement structure, templates, agreement states and the renegotiation protocol. A fuzzy support system to assess the benefits of the renegotiated SLA is also presented. Experiments are designed for an evaluation of dynamic SLAs through assessing the WS-Agreement extensions with the possibility of renegotiation at runtime. Put in the context of risk management, this evaluation shows that a service provider may increase profit and decrease the risk of SLA failure.

The work presented in this paper can be extended in many ways, which incude the introduction of a brokerage system, where the renegotiation protocol handles the interaction between the service consumer, the broker, and the service provider. The consideration of potential ripple effects of adopting new renegotiated SLAs at runtime is also subject of future investigation of the renegotiation aspects in Grids/Clouds, where infrastructures are by definition dynamic, and therefore the resources' status are subject to constant changes.

Acknowledgements

The authors would like to thank the European Commission for partly supporting this work under FP7-ICT-2009-5 contract 257115 Optimized Infrastructure Services (OPTIMIS).

References

1. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th*

utility, Future Gener. Comput. Syst. 25, 6 (Jun. 2009), 599-616.
2. E. Keller and H. Ludwig, *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*, Journal of Network and Systems Management, Vol. 11, pp. 57-81, 2003.
3. O. Waeldrich and W. Ziegler, *WSAG4J - Web Services Agreement for Java*, [online] Available: https://packcs-e0.scai.fraunhofer.de/wsag4j, July 3rd, 2012.
4. *BREIN: Business objective driven REliable and Inteligent grids for real busiNess*, 2007, http://www.eu-brein.com
5. P. Hasselmeyer, C. Qu, L. Schubert, B. Koller and P. Wieder, *Towards autonomous brokered SLA negotiation*, Proceedings of the 2006 eChallenges ConferenceExploiting the Knowledge EconomyIssues, Applications, Case Studies, Cunningham P, Cunningham M (eds.), vol. 3. IOS Press: Amsterdam, 2006. ISBN: 1-58603-682-3.
6. *Business Experiments in Grid*, 2009. http://www.beingrid.eu
7. *Grid-friendly software licensing for location independent application execution*, 2009. http://www.smartlm.eu
8. B. Rochwerger, A. Galis, E. Levy, J. Caceres, D. Breitgand, Y. Wolfsthal, M.W. Llorente, R. Montero and E. Elmroth *RESERVOIR: Management technologies and requirements for next generation service oriented infrastructures*, Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Management, New York, U.S.A., 2009. http://www.reservoir-fp7.eu
9. R. Kubërt, G. Gallizo, T. Polychniatis, T. Varvarigou, E. Oliveros, S.C. Phillips and K. Oberle, *Service Level Agreements for Real-time Service-Oriented Infrastructures*, Chapter 8 in Achieving Real-Time in Distributed Computing book: From Grids to Clouds. , IGI Global Books, Information Science Pub, May 2011.
10. G. Gallizo, R. Kuebert, K. Oberle, A. Menychtas and K. Konstanteli, *Service Level Agreements in Virtualised Service Platforms*, Proceedings of the eChallenges 2009 Conference (eChallenges 2009, Istanbul, Turkey, October 2008), IIMC International Information Management Corporation, 2009, ISBN: 978-1-905824-13-7.
11. G. Gallizo, R. Kuebert, G. Katsaros, K. Oberle, K. Satzke, S. Gogouvitis and E. Oliveros, *A Service Level Agreement Management Framework for Real-time Applications in Cloud Computing Environments*, In Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp 2010).
12. *Interactive RealTime Multimedia Applications on Service Oriented Infrastructures (IRMOS)*, 2011. http://www.irmosproject.eu
13. A. Menychtas, D. Kyriazis, S. Gogouvitis, K. Oberle, T. Voith, G. Gallizo, S. Berger, E. Oliveros and M. Boniface, *A Cloud Platform for Real-time Interactive Applications*, 1st International Conference on Cloud Computing and Service Science (CLOSER 2011), Noordwijkerhout, The Netherlands, May 7-9, 2011.
14. R. Kuebert, G. Gallizo, K. Oberle and E. Oliveros, *Enhancing the SLA Framework of a Virtualized Service Platform by dynamic re-negotiation*, Proceedings of the eChallenges 2010 Conference, Warsaw, Poland, October 27-29, 2010.
15. A. Pichot, P. Wieder, O. Wäldrich and W. Ziegler, *Dynamic SLA-negotiation based on WS-Agreement*, Technical Report CoreGRID TR-0082, June 2007.
16. G. Frankova, D. Malfatti and M. Aiello, *Semantics and Extensions of WS-Agreement*, Journal of Software, Vol 1, No 1 (2006), 23-31, Jul 2006
17. H. Ludwig, T. Nakata, O. Wäldrich and W. Ziegler Coregrid Tr, *Reliable Orchestration of Resources using WS-Agreement*, High Performance Computing and Communications Lecture Notes in Computer Science Volume 4208, pp 753-762, 2006.
18. P. Wieder, J. Seidel, O. Wäldrich, W. Ziegler and R. Yahyapour, *Using SLA for resource management and scheduling - a survey*, Workshop on Using Service Level Agreements in Grids, In Grid middleware and services: Challenges and solutions. Berlin: Springer US, 2008. (CoreGrid series 8), pp. 335-347.
19. M. Parkin, P. Hasselmeyer, B. Koller and P. Wieder, *An SLA Re-Negotiation Protocol*, In Proceedings of the 2nd Workshop on Non Functional Properties and Service Level Agreements in Service Oriented Computing at ECOWS 2008, Dublin, Ireland, November 2008.
20. G. Di Modica, V. Regalbuto, O. Tomarchio and L. Vita, *Enabling Re-negotiation of SLA by Extending the WS-Agreement specification*, IEEE International Conference on Services Computing (SCC'2007), Salt Lake City, USA, 9-13 July 2007.
21. G. Di Modica, O. Tomarchio and L. Vita, *Dynamic SLAs management in service oriented environments*, Journal Systems Software, Elsevier Science Inc., May, 2009, Vol. 82, pp. 759-771.
22. H. Ludwig, A. Dan, R. Kearney, *Cremona: An architecture and library for creation and monitoring of WSAgreements*, Proceedings of the International Conference on Service Oriented Computing (ICSOC'2004), pp.65-74, 2004. ACM: New York, NY, USA.
23. D. Neumann, J. Stoesser, A. Anandasivam, N. Borissov, *SORMA: Building an open market for grid resource allocation*, Grid Economics and Business Models (Lecture Notes in Computer Science, vol. 4685). Springer: Berlin, 2007; pp. 194-200.
24. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Kakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, *Web Services Agreement Specification (WS-Agreement)*, GRAAP-WG, OGF proposed recommendation, [online] Available: http://www.ogf.org/documents/GFD.192.pdf, October 10th, 2011.
25. B. Mitchell and P. McKee, *SLAs A Key Commercial Tool*, Exploiting the Knowledge Economy - Issues, Applications, Case Studies, eChallenges 2006.
26. A. Galati, K. Djemame, M. Fletcher, M. Jessop, M. Weeks, S. Hickinbotham, J. McAvoy, *Designing an SLA Protocol with Renegotiation to Maximize Revenues for the CMAC Platform*, at the Cloud-enabled Business Process Management (CeBPM 2012), November 28th-30th, 2012, Paphos, Cyprus.
27. *SLA@SOI: Empowering the service industry with SLA-aware infrastructures*, FP7-ICT-2007.1.2 Service and Software Architectures, Infrastructures and Engineering, [Online] Available: http://sla-at-soi.eu, Date of last access: July 4th, 2012.
28. D. Battré, F.M.T. Brazier, K.P. Clark, M.A. Oey, A. Papaspyrou, O. Wäldrich, P. Wieder and W. Ziegler, *A Proposal for WS-Agreement Negotiation*, In Proceedings of the 11th IEEE/ACM International Conference on Grid Computing, pp. 233-241, October 2010.
29. O. Wäldrich, D. Battré, F. Brazier, K. Clark, M. Oey, A. Papaspyrou, P. Wieder and W. Ziegler, *WS-Agreement Negotiation Version 1.0*, GRAAP-WG, Open Grid Forum, http://www.ogf.org/documents/GFD.193.pdf, October 2011.
30. I. Brandic, D. Music and S. Dustdar, *Service Mediation and Negotiation Bootstrapping as First Achievements To-*

wards Self-adaptable Grid and Cloud Services, In Grids and Service-Oriented Architectures for Service Level Agreements, P. Wieder, R. Yahyapour, and W. Ziegler (eds.), Springer, 2009.
31. K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, K. Voss and O. Kao. *Introducing Risk Management into the Grid*. Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam, The Netherlands, December 2006
32. K. Djemame, J. Padgett, I. Gourlay, and D. Armstrong, *Brokering of Risk-Aware Service Level Agreements in Grids*, Concurrency and Computation: Practice and Experience, Vol. 23, No. 7, May 2011
33. K. Hwang, G.C. Fox and J.J. Dongarra, *Distributed and Cloud Computing - From Parallel Processing and the Internet of Things*, Morgan Kaufmann, 2012
34. *Contract based e-Business System Engineering for Robust, Verifiable Cross-organisational Business Applications (CONTRACT)*, http://www.ist-contract.org, 2009
35. D. Battré, O. Kao and K. Voss *Implementing WS-Agreement in a Globus Toolkit 4.0 Environment*, Grid Middleware and Services (2008), Springer, pp. 409-418.
36. A. Juan Ferrer, F. Hernandez, J. Tordsson, E. Elmroth, C. Zsigri, R. Sirvent, J. Guitart, R.M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S.K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgo, T. Sharif, and C. Sheridan. *OPTIMIS: a Holistic Approach to Cloud Service Provisioning*. Future Generation Computer Systems, E1sevier, Vol. 28, No. 1, pp. 66-77, 2012.
37. Mathworks. *MATLAB: The Language of Technical Computing*. www.mathworks.co.uk/products/matlab, 2012
38. Z. Hua, B. Gong and X. Xu. *A* DS-AHP Approach for Multi-attribute Decision Making Problem with Incomplete Information. Journal of Expert Systems with Applications, Vol. 34, pp. 2221-2227, 2008