# A Proposal for WS-Agreement Negotiation

Dominic Battré*, Frances M.T. Brazier‡, Kassidy P. Clark‡, Michael Oey‡,
Alexander Papaspyrou§, Oliver Wäldrich†, Philipp Wieder¶, Wolfgang Ziegler†

*Technische Universität Berlin
Complex and Distributed IT Systems
10587 Berlin, Germany
dominic.battre@tu-berlin.de

†Fraunhofer SCAI
Schloss Birlinghoven
53754 Sankt-Augustin, Germany
{oliver.waeldrich, wolfgang.ziegler} @scai.fraunhofer.de

‡Systems Engineering
Faculty of Technology, Policy and Management
Delft University of Technology
The Netherlands
{f.m.brazier, k.p.clark, m.a.oey}@tudelft.nl

§Robotics Research Institute
TU Dortmund University
Dortmund, Germany
alexander.papaspyrou@tu-dortmund.de

¶IT Management and Service Computing Group
TU Dortmund University
Dortmund, Germany
philipp.wieder@udo.edu

***Abstract*—The Web Services Agreement specification defines a normative language to formulate Service Level Agreements and a basic protocol to expose service-level descriptions, validate service-level requests, and come to an agreement. This protocol, often called "take-it-or-leave-it", allows a service provider and a service consumer to decide whether to accept or reject a service offer. Although this approach is sufficient for a number of use cases, others exist with requirements for multi-step negotiation or the adaptation of an existing agreement.**

**In this paper, we describe the Web Services Agreement Negotiation protocol, a proposal 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, re-negotiation capability, and compliance with the original specification.**

***Keywords*-Negotiation Constraints, Open Grid Forum, Re-Negotiation, SLA Negotiation, WS-Agreement**

## I. Introduction

Service Level Agreements (SLAs) are one central pillar of Service Level Management solutions. In this area, industry best practices and standards like the TeleManagement Forum's SLA Handbook [1] or the IT Infrastructure Library [2] exist for quite some time and are adopted by various stakeholders. They cover the full life-cycle of Service Level Agreements, including steps like the agreement on Quality-of-Service terms and the resulting provision of the respective services.

One thing missing was the possibility to automatically negotiate SLAs that adhere to a normative format. The Web Services Agreement Specification (WS-Agreement [3]) provides one solution to close that gap by defining a normative language to formulate Service Level Agreements and a basic protocol to come to an agreement. With WS-Agreement being adopted by a diversity of stakeholders [4], it became evident that the basic negotiation protocol, as described in the related work section, does not fulfill the requirements of all stakeholder and usage scenarios since it realises a single shot process ("take-it-or-leave-it") without any negotiation. Therefore, the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG), the home of WS-Agreement at the Open Grid Forum (OGF[1]), gathered these requirements and came up

[1] http://www.ogf.org/

with a proposal for a SLA negotiation protocol called WS-Agreement Negotiation. In this paper, we describe the final draft version of this protocol.

The remainder of this paper is organized in the following way: We introduce work related to SLA negotiation in the next section and then describe the advance reservation use case, which gives one motivating example for the specification of WS-Agreement Negotiation. In Section IV, we show the negotiation model that is the basis of the actual protocol, which we describe in detail in Section V. Following this, we provide a number of examples on how to apply the protocol (Section VI) and then conclude with an outlook sketching the next steps necessary to make this proposal an OGF recommendation.

## II. Related Work

Negotiation is a widely studied topic and there are numerous publications addressing different aspects, e.g. [5] is a general purpose negotiation journal, [6] is a survey about negotiation in distributed resource management systems, while [7] and [8] discuss aspects of service negotiation in the Grid. In our context and in the simplest case, a user's job has to be executed and the Grid scheduler has to select between different target systems. If all systems are identical and only one parameter influences the selection, i.e. price, this case is similar to a typical business negotiation between one buyer and several sellers. An auctioning mechanism like the ones described in [9] can be used. Of course, we take the point of view of an end user, if we look at things from a resource provider's point of view, we have several jobs that compete for one resource, i.e. several buyers and one seller. If we look at the scheduler's point of view, we have many jobs that compete for several resources, i.e. many buyers and many sellers. Buyya [9] (page 36) also surveyed several distributed resource management systems based on price.

Another focus of research during the last years was on automatic negotiation of SLAs, which is a complex and time consuming process [10]–[12], when even two users have to find an agreement on multiple criteria. Imagine how difficult the problem becomes when multiple entities have to reach an agreement as presented one of the oldest approaches for SLA negotiation [13]. When at least two resources are needed at the same time to run a job, e.g. a network connection and a processing resource, several steps have to be performed before reaching an agreement between the resource providers and the consumer. However, even negotiating QoS and availability for a single resource can require several steps to reach an agreement. Green [12] cites mainly two frameworks for automatic negotiation: ontologies and web services. According to him automated negotiation has three main considerations: The negotiation protocol, the negotiation objects and the decision-making models. He considers two options existing in order to achieve this type of negotiation. One option is for the originating agent to negotiate separately with each Autonomous System (AS) along each potential path to ensure that an end-to-end path is available. The dominant choice however, is to use a cascaded approach where each AS is responsible for the entire path downstream of itself. This approach enhances agent autonomy as it is only responsible for its immediate links. The autonomy of the cascaded approach struggles however with the issue of price. In a cascading scenario an intelligent agent would need to know the utility functions of all the downstream domains if the best price combination is to be determined, which is private information. In contrast, in this paper we limited the scope to protocols that permit the negotiation of agreements between each two parties based on WS-Agreement rather than tackling the full complexity of automated negotiation. These individual bilateral agreements might then be combined into one single agreement. The approach for negotiation of agreements presented in this paper is aiming to provide a generic protocol that can be used in different scenarios, e.g. bilateral or multilateral negotiations, agent-based negotiations or auction-style negotiations.

A negotiation protocol suitable for service composition has been proposed by Jun Yan et al. [14]. Service Level Agreements for a service composition are established through autonomous agent negotiation. A framework is proposed in which the service consumer is represented by a set of agents who negotiate quality of service constraints with the service providers for various services in the composition. Based on this framework, the authors propose a new negotiation protocol to support coordinated negotiation. However, this approach does not consider the current WS-Agreement specification.

More recent research on SLA negotiation based on WS-Agreement is addressing unreliability of message transmission in distributed environments. Parkin et al propose a protocol based on the principles of contract law and capable to cope with the imperfect message transmission layer [15]. In [16] the authors present early research on extending WS-Agreement with negotiation capabilities taking into account the symmetrical approach WS-Agreement assumes for the roles of agreement initiator and agreement provider.

## III. The Advance Reservation of Compute Resources Use Case

In the scenario we select as an example, a service provider offering computing resources to customers. The computing resource service consists of a job submission service and portal application to manage the job submission service. The job submission service is a web service that provides methods for submitting and managing computing jobs, which are exposed via Web Service Description Language (WSDL) port types. The portal application provides methods to manage the job submission service, including user-related and resource-related operations, like updating a user profile or getting information about the resource consumption respectively.

Between the service provider and the consumers agreements are negotiated to have a shared understanding of Quality-of-Service provided and the obligations of either party. In the specific case here, it is of utmost importance to the consumer that the service provisioning starts at a certain point in time and lasts for a well-defined duration. To achieve this, the service provider offers the capability to reserve computing resources in

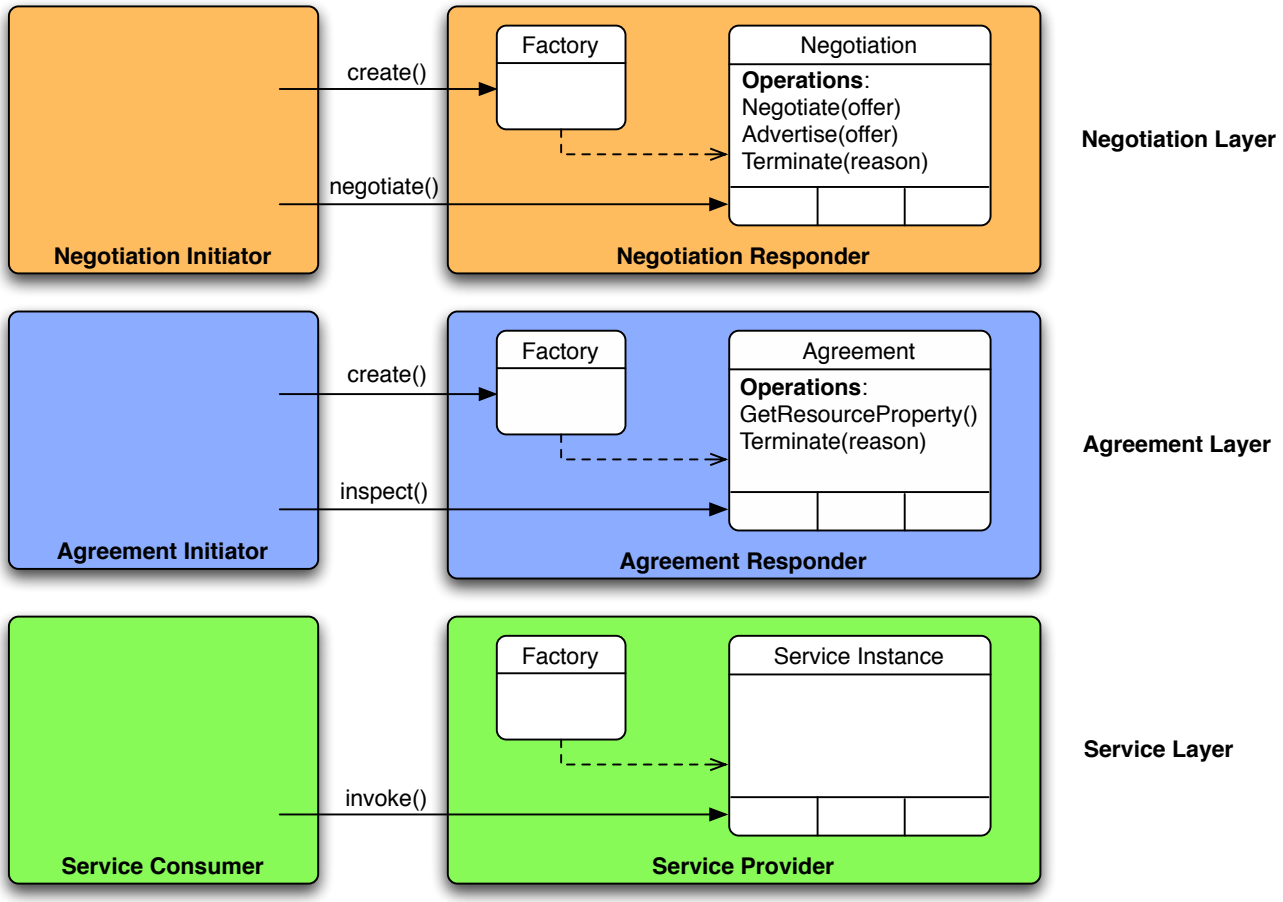


Fig. 1. Conceptual overview of the layered negotiation model

advance. In the case of the example the advance reservation is carried out through negotiation and governed by an agreement. The resource provisioning model is implementation-specific; whether resources are exclusively dedicated to a user, prediction models or preemption are used is up to the resource provider. Technically, the computing resource provider offers available computing resources via an agreement template [3]. The template includes the description of the service and its guarantees, as well as a set of options the customer can choose from. The service description contains the available computing resources and the time frame the resources are available (and hence can be reserved in advance). Moreover, the resource provider may offer different service levels for the computing resource service letting the customer e.g. choose between different classes of service availability (common values are specified as a the range between x% and y%) or average service response time (e.g. specified using a service-class name like gold, silver and bronze). The Quality-of-Service parameters can be specified separately for the job submission service and the portal application. The price of the service offer is then dependent on type and number of the selected computing resources and the respective service levels.

Potentially, the template provides many possibilities to parameterize the computing resource service. For example, the template contains parameters, such as pricing, that are dependent on the resources chosen and the service quality guaranteed. Once a customer has filled in all requirements into the SLA template, he sends the offer to the resource provider. The provider then checks whether the requested service can be provisioned at the requested time (i.e. whether the reservation of the resources can be carried out in advance). In case the service can be provided it sends back a completed counter-offer with the pricing information to the customer that in turn can now choose to create a negotiated agreement based on the offer. In case the resource provider is not able to fulfil all the requirements stated by the customer, it can also send back a counter-offer indicating a service quality it is able to provide instead. A customer has requested, for example, 128 nodes with 8GB memory in a given time frame, but the resource provider could not fulfil this request at this time. Instead the provider sends back a counter-offers for 96 nodes with 8GB memory and 32 nodes with 6GB memory for a lower price. The customer then may now choose to accept the counter-offer or any part of it purchasing the remaining resources somewhere else. The process of filling in all required fields of a negotiation offer can take multiple rounds.


Fig. 2. The Structure of a Negotiation Offer

At a later point in time (with an agreed upon SLA governing the job execution), the customer may recognise that he requires more resources to complete its computation. In that case the customer may initiate a re-negotiation of the agreement.

As there is currently no negotiation protocol that is compliant to WS-Agreement, offering multi-round negotiation and providing re-negotiation mechanisms, the GRAAP-WG specifies the negotiation protocol at hand.

## IV. The Service Level Agreement Negotiation Model

As described in Section II a number of different approaches and models for negotiation have been presented over the last couple of years. The GRAAP working group of the Open Grid Forum spent quite some time in evaluating the use of different approaches for the negotiation of SLAs. Moreover, the group also defined and evaluated a number of models since 2008. Finally, - after the decision to provide negotiation on top of WS-Agreement rather than modifying WS-Agreement to include negotiation - the negotiation model presented in this section was developed and refined during several working group meetings at the Open Grid Forum and several dedicated small workshops. Major requirements for the development process were compliance with WS-Agreement, support for bilateral and multilateral negotiations, domain independency allowing to use the protocol in different environments, e.g. auctions or agent-based infrastructures, the possibility to later easily exchange the protocol against another one if more (specific) negotiation protocols evolve. Finally, the protocol should be symmetric with respect to the parties negotiating, service provider and service consumer. Details of the discussion process and how the protocol evolved can be found in the working group's Gridforge Project[2].

The WS-Agreement negotiation model consists of three layers, the negotiation layer, the agreement layer and the service layer. These layers are depicted in Figure 1.

There is a clear separation between these three layers in the negotiation model. The negotiation layer sits on top of the agreement layer. Therefore, it is decoupled from the agreement layer and the service layer. By that, the negotiation layer may change independently of the agreement layer and be replaced by another negotiation layer that may be better suited for specific negotiation scenarios. Agreement layer and service layer are modelled according to the WS-Agreement specification [3].

### A. Negotiation Layer

The negotiation layer provides a protocol and a language to negotiate agreement offers and counter-offers, and to create agreements based on negotiated offers. The negotiation process comprises the exchange of offer and counter-offers. Negotiation offers, as defined in Section V-C, are non-binding by nature. Therefore, negotiated offers do not make any promises that a subsequent agreement based on a negotiated offer will be created. They only indicate the willingness of two negotiating parties to accept a subsequent creation of an agreement. However, WS-Agreement Negotiation can be extended to realize binding negotiation processes, but this is not in the focus of the current work and specification.

Once no further (counter-)offers are requested by one of the parties, negotiated agreements are then created by calling either the `createAgreement` or the `createPendingAgreement` operation on the agreement layer's Agreement Responder Factory instance (see [3] for details on WS-Agreement's operations).

### B. Agreement Layer

The agreement layer provides the basic functionality to create and monitor agreements. It provides a protocol and a language defined in the WS-Agreement specification, to which we refer for further details.

### C. Service Layer

At the service layer the actual service defined by an agreement is provided. This service may or may not be a web service. Moreover, a service defined by an agreement may consist of multiple services, e.g. a service for resource provisioning may consist of the provisioning service and a monitoring service for the provided resources. The service execution on the service layer is governed by the agreement layer.

## V. WS-Agreement Negotiation

A negotiation (as depicted in Figure 1) is a service instance that is used by two negotiating parties to exchange information in order to come to a common understanding of valid agreement offers. In the process of a negotiation the two

[2]https://forge.gridforum.org/sf/projects/graap-wg

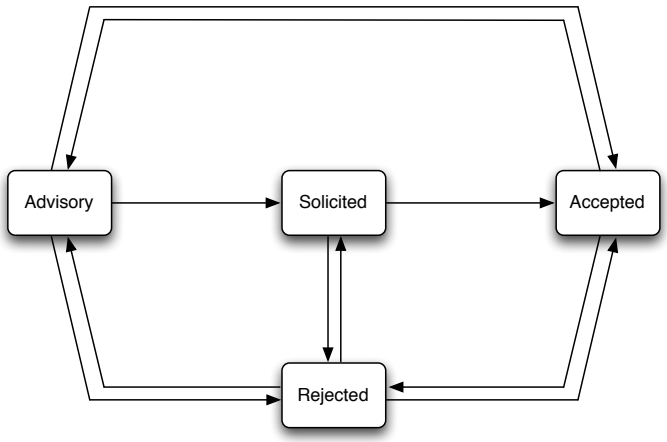


Fig. 3. Negotiation State Machine

negotiating parties exchange negotiation offers and indicate their goals and requirements. A negotiation may be limited in lifetime or rounds of negotiation. These limitations are defined in the negotiation context as described in Section V-B. In the following sections the key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' are to be interpreted as described in RFC 2119 [17].

### A. Protocol

The protocol is providing support for bilateral negotiation of agreement offers. It is designed as a symmetric protocol, thus maintaining the inherent symmetry of the basic WS-Agreement protocol. In fact, it was an explicit design goal to support full symmetry regarding rights, obligations and capabilities of service consumer and service provider.

### B. Negotiation Context

The negotiation context defines the roles of the negotiation participants, their obligations, and the nature of the negotiation process. Since a negotiation is a bi-lateral process, the roles of each participating party must be clearly defined. In general a negotiation process can either refer to the negotiation of new agreements or the re-negotiation of an existing agreement. Therefore, the type of the negotiation must be defined in the negotiation context. Moreover, the negotiation context defines the roles of the parties participating in the negotiation process. The negotiation participants must acknowledge these parameters for the entire negotiation process. The following listing shows a pseudo-schema[3] of the elements included in the negotiation context:

```
<wsag-neg:NegotiationContext>
    <wsag-neg:NegotiationType>
        wsag-neg:NegotiationType
    </wsag-neg:NegotiationType>
    <wsag-neg:ExpirationTime>
        xsd:dateTime
    </wsag-neg:ExpirationTime> ?
    <wsag-neg:NegotiationInitiator>
    xsd:anyType
    </wsag-neg:NegotiationInitiator> ?
    <wsag-neg:NegotiationResponder>
          xsd:anyType
    </wsag-neg:NegotiationResponder> ?
    <wsag-neg:AgreementResponder>
          wsag-neg:NegotiationRoleType
    </wsag-neg:AgreementResponder>
    <wsag-neg:AgreementFactoryEPR>
          wsa:EndpointReferenceType
    </wsag-neg:AgreementFactoryEPR> ?
</wsag-neg:NegotiationContext>
```

The *NegotiationType* specifies the nature of the negotiation process, i.e. either the negotiation of a new agreement or the re-negotiation of an existing agreement. In the latter case, a reference to the agreement that is re-negotiated is provided.

Independent of the type of negotiation, every negotiation instance may carry an OPTIONAL *ExpirationTime*, which specifies the lifetime of the negotiation instance. If specified, the negotiation instance is accessible until the given time. After its lifetime, the negotiation instance is no longer accessible. For the *ExpirationTime* it is beneficial for the consistent interpretation of the *ExpirationTime* by the different parties if their clocks are synchronised as today usually realised with the Network Time Protocol (NTP).

*NegotiationInitiator* This OPTIONAL element identifies the initiator of the negotiation process. The negotiation initiator element can be an URI or an Endpoint Reference that can be used to contact the initiator. It can also be a distinguished name identifying the initiator in a security context.

*NegotiationResponder* This OPTIONAL element identifies the party that responds to the initiation of the negotiation process. This party implements the NegotiationFactory port type of this specification. This element can be an URI or an Endpoint Reference that can be used to contact the negotiation responder. It can also be a distinguished name identifying the negotiation responder in a security context.

*AgreementResponder* This REQUIRED element identifies the party in the negotiation process that acts on behalf of the agreement responder. This element can either take the value NegotiationInitiator or NegotiationResponder. The default value is NegotiationResponder, The party identified as agreement responder MUST provide a reference to the AgreementFactory (PendingAgreementFactory) it acts on the behalf of in the negotiation context by using the AgreementFactoryEPR element.

*AgreementFactoryEPR* This OPTIONAL element identifies the endpoint reference of the agreement factory that SHOULD be used to create agreements based on the negotiated agreement offers. After an agreement offer was successfully nego-

[3]The pseudo-schema uses BNF-style conventions for attributes and elements: '?' denotes zero or one occurrences, '*' denotes zero or more occurrences, and '+' denotes one or more occurrences.

tiated, the party identified as agreement initiator MAY create a new agreement with the referenced factory.

A negotiation process comprises the exchange of offers and counter-offers. Counter-offers are created based on existing offers. An initial offer is created on the basis of an Agreement Template. The structure of a negotiation offer is basically the same as the structure of an Agreement Template. Agreement templates are defined in the Agreement Template and Creation Constraints section of the WS-Agreement specification. However, a negotiation offer contains the additional elements Negotiation Offer Context and Negotiation Constraints.

### C. Structure of an Offer

In order to negotiate the content of an agreement, negotiation offers are exchanged between an agreement initiator and an agreement responder. In case one of the negotiating parties receives a negotiation offer, this party evaluates the offer and creates zero or more counter-offers, which are then sent back to the negotiation participant. The basic structure of a negotiation offer is shown in Figure 2.

A negotiation offer has basically the same structure as an agreement template [3], but in addition, the negotiation offer contains two more elements: a Negotiation Offer Id and a Negotiation Context (see Section V-B).

A Negotiation Offer also contains a Negotiation Constraints section. The Negotiation Constraints define the structure, valid ranges or distinct values that Service Terms may take in a counter-offer. The Negotiation Constraints of a negotiation offer must hold true for every counter-offer. In a negotiation process, however, the Creation Constraints MAY change during the advance of the negotiation. For example, if the negotiation initiator chooses one specific Service Term out of a set of Service Terms (ExactlyOne), a negotiation responder may adopt to this choice by changing the Creation Constrains section in a Counter Offer.

Negotiation Constraints are structurally identical to the Creation Constraints defined in an agreement template. Creation Constraints are defined in the section Agreement Template and Creation Constraints of the WS-Agreement specification.

### D. Negotiation Offer States

During the negotiation process the content of Agreement Offers is negotiated before an agreement is created. A negotiated agreement is created by the party identified as Agreement Initiator in the negotiation context. A valid negotiated agreement offer MUST have the state Agreed when a new negotiated agreement is created. Figure 3 illustrates the states that negotiation offers can have and the valid state transitions.

*1) Advisory State:* The Advisory State identifies negotiation offers with no further obligations associated with. Offers in the Advisory State usually contain elements that are currently not specified. Therefore, these offers require further negotiation.

*2) Solicited State:* The Solicited State bears no obligations for an offer, but it requires that counter offers are either in the Accepted or the Rejected State. Solicited offers indicate that a negotiation participant wants to converge the negotiation process and requests only counter offers that can be accepted as is, e.g. where no further negotiation of the counter offers is required.

*3) Accepted State:* The Accepted State indicates that a negotiation participant accepts a negotiation offer as is. All details of a negotiation offer are specified and no further negotiation is required. However, since the negotiated offers are non-binding, there is no guarantee that a subsequent agreement is created. Augmented negotiation protocols may be created based on this specification to address binding negotiations.

*4) Rejected State:* If a negotiation offer is rejected, it is sent back to the inquiring party with the rejected state. The negotiation offer MAY contain a domain specific reason why it was rejected. Negotiation offers that are marked as rejected MUST NOT be used to create an agreement. However, they MAY be used to continue the negotiation process by taking into account the reason for rejecting the offer.

## VI. Different Negotiation Examples

In this section a detailed description of the Negotiation Factory and the Negotiation port types is given through a set of example scenarios. These port types can be used in different combinations in order to support a wide range of signalling scenarios. The presented signalling scenarios are not meant to cover all possible combinations of the port types. They are presented here to illustrate possible negotiation scenarios and how these scenarios are mapped to specific deployments of WS-Agreement Negotiation. Furthermore, the interaction of the negotiation layer and the agreement layer is discussed.

### A. Basic Client–Server Example

The simple client-server negotiation is an asymmetric signalling scenario, where a server implements the Negotiation Factory and Negotiation port types. The negotiation process itself is driven by the client. The client initiates a negotiation by calling the server's `initiateNegotiation` operation of the Negotiation Factory. After a new negotiation is created, the client queries the available templates from the Negotiation Responder that serve as initial templates for a negotiation offer. It uses these templates to create new negotiation offers and sends these offers to the server via the negotiate method of the Negotiation port type. The server may create one or more counter-offers for each offer that is part of the negotiate call. The server itself has a passive role in this negotiation process since it cannot actively influence the course of a negotiation, i.e. it can only react to negotiation requests. The process of negotiation is depicted in Figure 4.

### B. Bilateral Negotiation with Asymmetric Agreement Layer

In a bilateral negotiation both parties can actively participate in the negotiation process. Both parties implement the WS-Agreement Negotiation port types. The process of initiating a bilateral negotiation is as follows. The Negotiation Initiator creates a new negotiation instance that implements the WS-Agreement Negotiation port type. It then sends an `initiateNegotiation` request

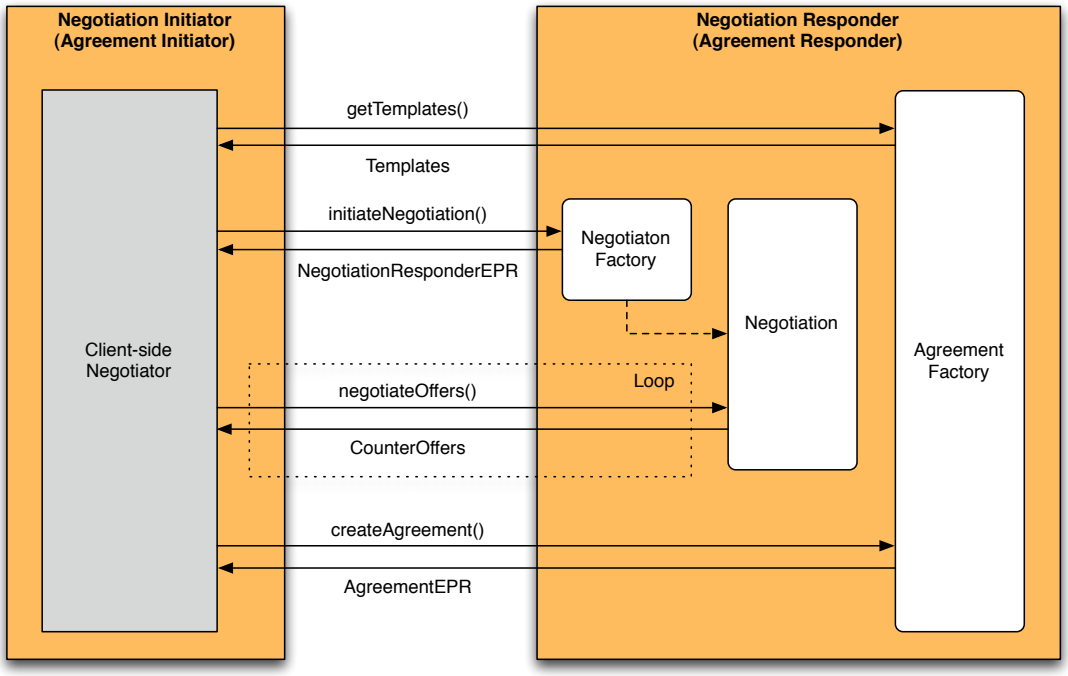


Fig. 4. Basic Client–Server Example showing Negotiation Layer and Agreement Factory (Asymmetric Deployment of the WS-Negotiation Port Types)

to the NegotiationFactory of the Negotiation Responder. The `initiateNegotiation` request includes an endpoint reference to the negotiation instance that was created beforehand. Moreover, it contains the negotiation context that defines the roles of each party in a negotiation, e.g. which party is the initiator and which is the responder of the agreements that are negotiated. In a bilateral negotiation process, the agreement templates that are used to create offers are provided by the agreement factory referenced in the negotiation context. The agreement initiator should query the available agreement templates from the agreement factory in order to create negotiation offers based on the provided templates. After a new Negotiation instance was created, the context of a negotiation must not change. Both parties participating in a negotiation process may actively send negotiation requests to the other party. It is not required that the initiator of a negotiation is also the initiator of the subsequent agreement. These roles may vary in different negotiation scenarios.

In the negotiation scenario depicted in Figure 5 the negotiation initiator is also the initiator of the subsequent agreements. It starts a negotiation process by retrieving the templates provided by the responder. Then the initiator notifies the responder of the offers it is willing to negotiate by calling the responder's `advertise` method. Now the negotiation responder takes an active role in the negotiation process by sending offers to the initiator. After several rounds of negotiation the initiator may decide to create an agreement based on one of the negotiated offer. It therefore calls the `createAgreement` method of the negotiation responder. The input of the `createAgreement` operation includes a critical extension that is the context of the negotiated offer that the initiator uses to create an agreement.

### *C. Re-Negotiation of Agreements with Symmetric Agreement Layer*

In general the re-negotiation of an existing agreement follows the same signalling pattern as the negotiation of an agreement. If an existing agreement is re-negotiated, the initiator of the original agreement should match the initiator of the re-negotiated agreement, so that the roles and obligations match the original agreement (see Figure 6). The roles and the responsibilities of the negotiating parties are defined in the negotiation context, when a new negotiation is created. The negotiation context also includes an endpoint reference to the existing responder agreement. In a symmetric signalling scenario, the negotiation context may additionally include a reference to the original initiator agreement. After a new re-negotiation process has been initiated, both parties start to negotiate the contents of the agreement offer that can be used to create a re-negotiated agreement. When they succeeded to negotiate a suitable offer, the initiator of the negotiated agreement creates a new agreement by invoking the `createAgreement` (`createPendingAgreement`) method of the responders Agreement Factory (Pending Agreement Factory) instance. When a re-negotiated agreement is created, the original agreement must transition into the COMPLETED state.

The layout of the agreement layer may either be asymmetric or symmetric. In case of a symmetric layout of the agreement layer, the re-negotiated agreement

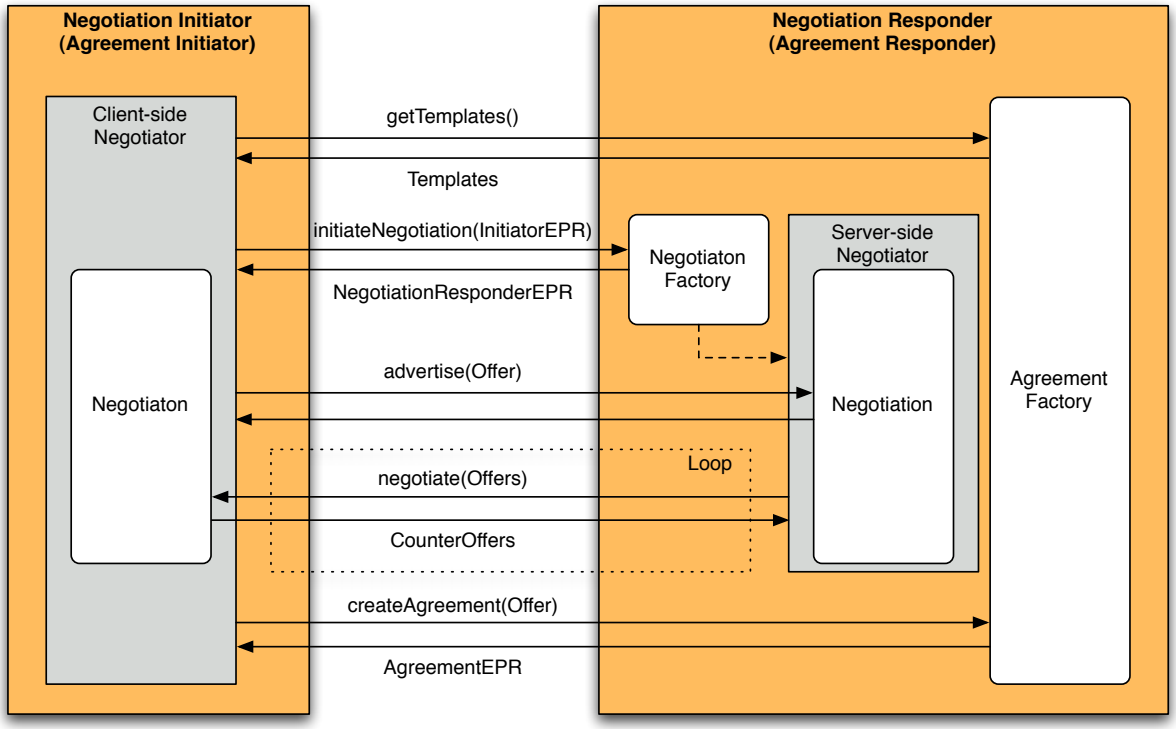


Fig. 5. Bilateral negotiation (Symmetric deployment of WS-Agreement Negotiation, where the Negotiation Initiator is also the Agreement Initiator and the Negotiation Responder is the Agreement Responder. Both Parties have an Active Role in the Negotiation Process.)

initiator creates an instance of the re-negotiated agreement before the `createAgreement` method (`createPendingAgreement` method) of the responder's agreement factory instance is invoked. This agreement must be in PENDING state until the responder has either accepted or rejected the creation of the re-negotiated agreement. After the initiator received the agreement responder's decision, the state of the pending agreement is updated accordingly. When a re-negotiated agreement is accepted, both parties must update the state of their original agreement instance to COMPLETED. Differences in the state of the original and re-negotiated agreements are handled in domain specific manner, e.g. by applying state replication, different levels of escalation or dispute handling.

## VII. Discussion and Outlook

We described a negotiation protocol for Service Level Agreements that offers multi-round negotiation and re-negotiation capabilities. Furthermore, it is compliant with the WS-Agreement proposed recommendation and does not require modifications of the WS-Agreement specification v1.0. The protocol proposed has been under discussion in the GRAAP working group of the Open Grid Forum and is now ready to be submitted to the public comment process of the Open Grid Forum. The authors expect the current draft including eventual modifications resulting from the public comment period to become a proposed recommendation of the Open Grid Forum by end of 2010.

The work for defining the next version of WS-Agreement decoupling of the basic protocol from the language for creating agreements along with minor improvements will start by end of 2010. This major change will render WS-Agreement more flexible allowing to negotiate and create agreements with domain specific protocols without sacrificing compatibility.

## VIII. Acknowledgements

Some of the work reported in this paper has been funded by the European Commissions ICT programme within the FP6 CoreGRID Network of Excellence under grant #004265 and in the FP7 project SmartLM under grant #216759. This paper also includes work funded by the German Federal Ministry of Education and Research through the D-Grid project under grant #01AK800A. Last not least the authors gracefully acknowledge the stimulating discussions in the GRAAP-WG.

## References

[1] T. T. Forum, "SLA Management Handbook, Volume 2, Concepts and Principles," Heidelberg, Germany, 2005, release 2.5.

[2] R. Addy, *Effective IT Service Management – To ITIL and Beyond!* Morristown, New Jersey, United States: Springer-Verlag: Berlin, 2007.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," Open Grid Forum, Grid Forum Document GFD.107, 2007.

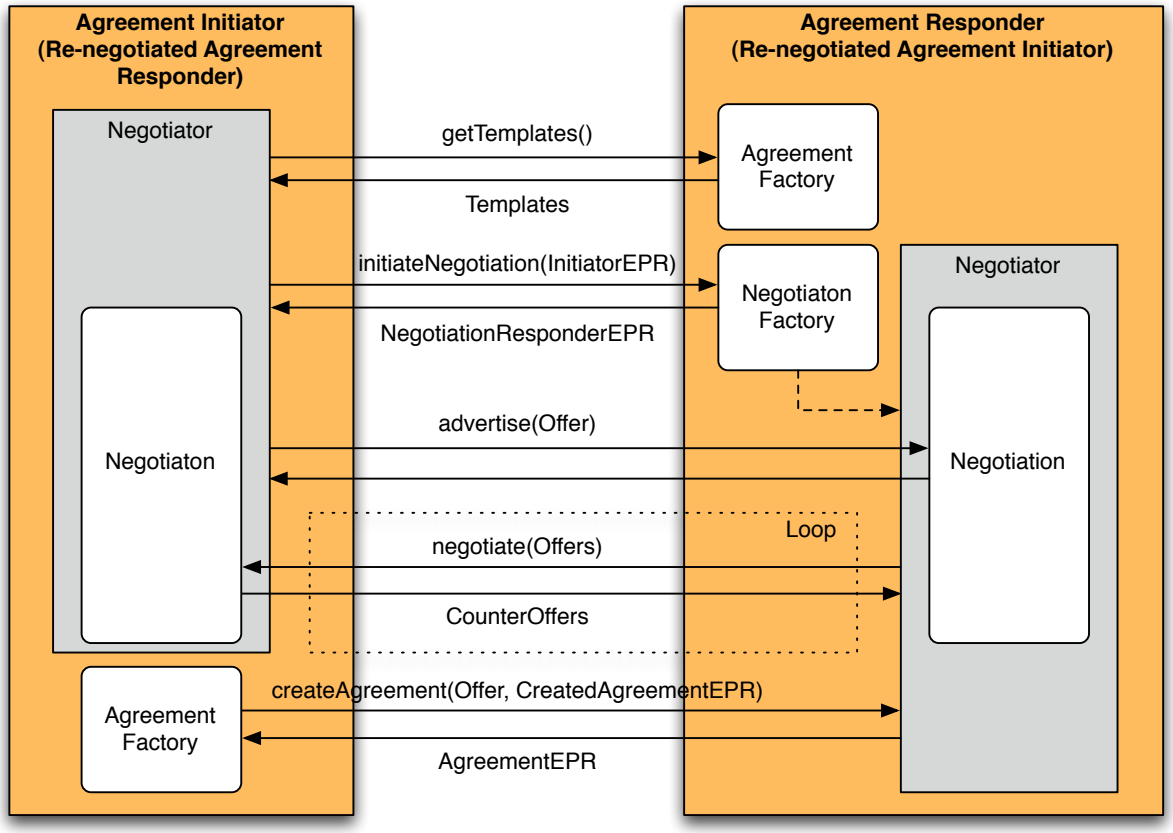


Fig. 6. Re-Negotiation of Agreements (Symmetric signalling on the Negotiation and Agreement Layer. Both parties implement the WS-Agreement Negotiation and WS-Agreement port types. Moreover, both parties have their own instance of the original agreement. After the negotiation process, the responder of the original agreement creates the re-negotiated agreement.)


[4] J. Seidel, O. Wäldrich, P. Wieder, R. Yahyapour, and W. Ziegler, "SLA for Resource Management and Scheduling - A Survey," in *Grid Middleware and Services: Challenges and Solutions (Proceedings of the Usage of Service Level Agreements in Grids Workshop in conjunction with the 8th IEEE International Conference on Grid Computing (Grid 2007))*, ser. CoreGrid series 8, D. Talia, R. Yahyapour, and W. Ziegler, Eds. Springer US, 2008.

[5] M. Shakun, Ed., *Group Decision and Negotiation*. Springer Netherlands, 2002.

[6] C. Briquet and P.-A. de Marneffe, "Grid resource negotiation: survey with a machine learning perspective," in *PDCN'06: Proceedings of the 24th IASTED international conference on Parallel and distributed computing and networks*. Anaheim, CA, USA: ACTA Press, 2006, pp. 17–22.

[7] D. Kuo, M. Parkin, and J. Brooke, "Negotiating Contracts on the Grid," in *Exploiting the Knowledge Economy - Issues, Applications, Case Studies, Volume 3, Proceedings of the eChallenges 2006 (e-2006) Conference*. Amsterdam, The Netherlands: IOS Press, 2006.

[8] ——, "A Framework & Negotiation Protocol for Service Contracts," in *Proceedings of the 2006 IEEE International Conference on Services Computing (SCC 2006)*, 2006, pp. 253–256.

[9] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis*. Melbourne, Australia: Monash University, 2002.

[10] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, "Automated Negotiation: Prospects, Methods and Challenges," *Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, March 2001.

[11] W. Shen, H. H. Ghenniwa, and C. Wang, "Adaptive Negotiation for Agent-Based Grid Computing," in *Proceedings of AAMAS2002 workshop on agentcities: Challenges in Open Agent Environments*, Bologna, Italy, 2002, pp. 32–36.

[12] L. Green, "Service level negotiation in a heterogeneous telecommunication environment," in *Proceeding International Conference on Computing, Communications and Control Technologies (CCCT04)*, Austin, TX, USA, August 2004.

[13] K. Czajkowski, I. T. Foster, C. Kesselman, V. Sander, and S. Tuecke, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems," in *JSSPP '02: Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK: Springer-Verlag, 2002, pp. 153–183.

[14] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. K. Goh, and J. Zhang, "Autonomous service level agreement negotiation for service composition provision," *Future Gener. Comput. Syst.*, vol. 23, no. 6, pp. 748–759, 2007.

[15] M. Parkin, P. Hasselmeyer, and B. Koller, "An SLA Re-Negotiation Protocol," 2008, proceedings of the 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop NFPSLA-SOC'08.

[16] W. Ziegler, P. Wieder, and D. Battré, "ExtendingWS-Agreement for dynamic negotiation of Service Level Agreements," Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, Tech. Rep. TR-0172, August 2008. [Online]. Available: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0172.pdf

[17] S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119." March 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt