Dynamic SLAs management in service oriented environments

Giuseppe Di Modica, Orazio Tomarchio*, Lorenzo Vita

Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Università di Catania, Viale A. Doria 6, 95125 Catania, Italy

ABSTRACT

The increasing adoption of service oriented architectures across different administrative domains, forces service providers to use effective mechanisms and strategies of resource management in order for them to be able to guarantee the quality levels their customers demands during service provisioning. Service level agreements (SLA) are the most common mechanism used to establish agreements on the quality of a service (QoS) between a service provider and a service consumer. The WS-Agreement specification, developed by the Open Grid Forum, is a Web Service protocol to establish agreements on the QoS level to be guaranteed in the provision of a service. The committed agreement cannot be modified during service provision and is effective until all activities pertaining to it are finished or until one of the signing party decides to terminate it. In B2B scenarios where several service providers are involved in the composition of a service, and each of them plays both the parts of provider and customer, several one-to-one SLAs need to be signed. In such a rigid context the global QoS of the final service can be strongly affected by any violation on each single SLA. In order to prevent such violations, SLAs need to adapt to any possible needs that might come up during service provision. In this work we focus on the WS-Agreement specification and propose to enhance the flexibility of its approach. We integrate new functionality to the protocol that enable the parties of a WS-Agreement to re-negotiate and modify its terms during the service provision, and show how a typical scenario of service composition can benefit from our proposal.

© 2008 Elsevier Inc. All rights reserved.

1. Introduction

The extensive usage of information and communication technology in modern organizations affects B2B interactions too. While traditionally, the interaction has mainly been based on manual processes, the use of digital means allows organizations to increase both the effectiveness and efficiency of their mutual interaction and consequently improves their business goals. Such interactions must be formalized to clearly define rights and obligations of all the involved parties. The formalization usually takes the form of a contract that defines the parties engaging in the collaboration, the goods, the services or funds exchanged between parties, and details about the way this exchange takes place. The use of electronic contracts with automated support for their management allows an increase of effectiveness and efficiency in contract processing, opening new possibilities for interaction among parties (Angelov and Grefen, 2001, 2004).

This issue becomes more important in service oriented architectures (SOA) (Papazoglou, 2003, 2007), which have recently appeared as an emerging paradigm for automated business integration. In this context the term “service” refers to a software resource, which is a well defined, self-contained module that provides standard business functionality and is independent of the state or the context of other services. When a customer wants to use a service offered by a Service Provider, an agreement is needed, in the same way of a traditional service. A service level agreement (Overton, 2002) is though a formal negotiated agreement between a Service Provider and its customer (the service requester). When a customer requests a service to a provider, an SLA is negotiated and a contract is drawn up. The contract involves both functional and not functional parameters relating to the service to be provided. The high level of interoperability offered by the SOA enables scenarios of world-wide and cross-domains service composition. The end-to-end management of SLAs in these scenarios is a tricky task. In a service-oriented scenario a service may be the result of the composition of several services deployed in as many administrative domains, each of which autonomously manages resources that are quantitatively and qualitatively diverse. In such a scenario, where the resource availability is highly dynamic, promising and guaranteeing specific QoS levels to the customer is a real challenge. The quality of the final service delivered to the customer is strongly affected by those services employed to compose it. If just one of the composing services violates at run-time its QoS guarantees, the global QoS delivered to the customer might get definitively compromised, to the detriment of the customer. There is a need for a flexible mechanism enabling the run-time re-negotiation of the guarantees on the QoS once violations on such guarantees are expected to occur.
In order to describe the contract between the Service Provider, the customer and an eventual third party, several specifications, often defined by XML-based languages, have been recently proposed. Many of them are a complement to the service description implemented by WSDL (Web services description language, 2006) and are used within a framework allowing the management of Web Services and their compositions. Our attention has been focused on the WS-Agreement (Andrieux et al., 2007) specification, that has been developed by the GRAAP Working Group (Grid Resource Allocation and Agreement Protocol WG) of the Open Grid Forum (OGF). It does not deal with negotiation mechanisms or protocols, rather it proposes to standardize the activities that follow the negotiation phase, i.e., those involving the signing of a contract between the parties.

In this work we propose to enhance the current WS-Agreement specification in order to introduce a run-time support to the renegotiation of the guarantees on the QoS levels as they were agreed in the contract by the signing parties. We do not add a negotiation layer, rather we propose a mechanism that enables re-negotiation sessions between the parties and the subsequent modification of the original contract. An implementation prototype has been also developed and it has been validated by building a simple service composition scenario. The paper is organized in the following way. Sections 2 and 3 present the motivations for this work together with an example scenario. The WS-Agreement specification and its limitation are described in Section 4. Section 5 presents our proposal, including some implementation details. Related works are described in Section 6, and finally we conclude our work in Section 7.

2. Rationale

In a B2C scenario, after the negotiation phase, the two parties involved in a transaction (namely the Service Consumer and the Service Provider) agree upon the Service Level Objectives (SLO) that the service provision must attain by signing a service level agreement (SLA) document.

In a typical B2B scenario the provision of a service may often require a form of collaboration among several Service Providers across multiple independent and heterogeneous administrative domains. The objective of a complex service provision can be accomplished by adopting plans of composition of heterogeneous services, whose single quality level contributes to the overall quality of the final service to be delivered to the customer. In such a scenario, several one-to-one SLAs are signed by the involved parties contributing to the provision of the final service to the end customer. The overall QoS perceived by the customer strictly depends on the fulfillment of each of these SLAs.

In a B2C scenario very often, because of inability of a Service Provider’s to meet an SLA’s service level objectives, the provision of the service that the SLA applies to is brutally stopped, and the SLA is terminated. In a B2B context much higher is the probability that the provision of a service to the customer is not successfully carried out. The higher the number of services needed to serve the Customer’s request, the higher the probability that the task is not accomplished.

For instance, let us consider the case of Customer C that wants to use a service offered by Service Provider SP1, which in its turn needs to access a service offered by Service Provider SP2 to provide its own service to the Customer. Again, let us assume that the service offered by SP2 is built on top of another service provided by SP1. Each service provision requires the sign of an SLA. The result here is that every provider of services play the role of Service Provider in a SLA and Service Customer in another one. In Fig. 1 a graphic representation of this scenario is given:

The involved parties (the customer C and the Service Providers \(SP_1, SP_2, SP_3\)) are represented by nodes, while the oriented edges represent the SLAs signed between pairs of parties and the respective roles that the parties play in the SLAs. For instance, the edge \(SLA_{23}\), originated from \(SP_1\) to \(SP_2\), represents the SLAs signed by the parties \(SP_1\) and \(SP_2\), being \(SP_1\) the Service Provider and \(SP_2\) the Service Consumer for that specific contract. The abnormal termination of a given agreement may cause an interruption in the SLAs’ chain, thus preventing the forwarding SLAs from being fulfilled. For instance, if \(SP_2\) for any reason could not attain the promised QoS target (or worse, stopping providing its service to \(SP_3\)), thus violating \(SLA_{12}\), \(SP_2\) might not be able to offer its service to \(C\) and then to honor \(SLA_{1-C}\). In this case, \(SP_2\) has two alternatives: either it finds another service provider offering the same service and the same guarantees as those offered by \(SP_2\), or it terminates \(SLA_{12}\) (thus incurring penalties). Furthermore, \(SP_2\) might not need to access the service offered by \(SP_3\) anymore (unless that service is useful to satisfy other pending requests), and will likely decide to terminate \(SLA_{3}\). We can conclude that in the worst case, the violation of the guarantees for an SLA may cause the termination of all the SLAs in the chain (three in this specific scenario).

In the same scenario of service composition, let us now assume that \(C\) is not satisfied with the perceived quality of the service being offered by \(SP_1\), and decide to negotiate with it for an improvement of the service’s quality level. Though \(SP_2\) has enough inbound resources to satisfy \(C’\)s request, an increment of the capacity of the service offered by \(SP_2\) is necessary to support the newly required overall QoS. Thus \(SP_2\) negotiates with \(SP_3\) to obtain a better level of QoS. On its turn, \(SP_3\) will not be able to provide the extra capacity unless \(SP_3\) is willing to sustain a higher level of quality of the service it provides to \(SP_2\). The original request of \(C\) can be satisfied only if the negotiations for the increment of service capacity offered by each provider are successful. In that case, the three SLAs can be modified and the quality of the services are adjusted to the new agreed levels.

In a more complex scenario, like that depicted in Fig. 2, in which several service providers are involved in the building of a service for the Customer, the probability of violation of the SLA regulating that service is very high.

If, for instance, \(SP_6\) for any reason was not able to honor \(SLA_{6}\), such a violation might propagate through the chain of agreements and might get to \(SP_2\), thus affecting \(SLA_{1-C}\). The risk of violation is high not only for the SLAs in the chain that links \(SP_6\) to \(C\), but also...
for the rest of the SLAs. Summarizing, in a service composition tree, the “breaking off” of an edge (SLA) linking any two nodes (service providers) may propagate throughout the tree affecting any other edge. As long as an SLA is a static and inflexible contract that regulates a transaction between two parties, the delivery of a service in service-composition scenarios involving multiple one-to-one SLAs among parties has a few chances to be successfully carried out.

Furthermore, given the number of Service Providers and SLAs involved in the scenario, any single request of improvement of the quality of the service coming from the Customer strongly relies has chances to be accommodates only if there is the possibility of modifying on-the-run the service level agreements signed by the involved parties.

A new concept of SLA need to be adopted. An SLA should be a dynamic contract capable of reacting and adapting to changes in the context it works on. A high degree of flexibility is needed in order to reduce, during the provision of the service that the contract applies to, any possible risk of violation of the QoS guarantees agreed by the parties at the time of the contract signing. The flexibility that we refer to consists in the possibility of (1) re-negotiating at run-time (i.e., during the service provision) the QoS guarantees, (2) accordingly modifying the terms of the contract, and (3) adjusting the service provision to the newly agreed QoS levels. This process, of course, must preserve the continuity of the service provision, i.e., the service flow must not be either interrupted or suspended while the contract is being re-negotiated and modified.

3. A scenario of service composition

Let us picture a scenario where a Video on Demand provider VoDpro is offering services in the Administrative Domain AD1 managed by the Domain Manager DM1. Let us also suppose that a Customer C, who is given connectivity in a different Administrative Domain AD2 managed by the Domain Manager DM2, wants to access the services offered by VoDpro. Fig. 3 depicts the actors of the scenario, the two Administrative Domains and their adjacency links.

C will have to specify the kind of the service he desires to use (i.e., multimedia streaming) and will have to negotiate the desired QoS target (expressed in terms of frame size, frame rate, bit rate, color depth and codecs).

VoDpro takes in consideration C’s desired QoS and decides to either accept or reject the request depending on the availability of the resources needed to sustain it. Two kind of resources are needed to serve C’s request: inbound resources (such as the VoD servers), that are entirely under the control of VoDpro, and out-bound resources (such as the connection path from the VoD servers to the C). In order to reserve the connection path VoDpro will have to negotiate with the managers of the connection resources (e.g., the bandwidth). As the path is composed of two different paths, traversing, respectively, the domains AD1 and AD2, two different negotiations are needed. In this case, to set-up a suitable connection path to C, VoDpro negotiates with DM1 the reservation of the required bandwidth within AD1; in its turn, DM1 negotiates with DM2 to reserve the required bandwidth within AD2 (this case can be reconducted to the “chain” configuration as depicted in Fig. 1). Alternatively, VoDpro itself may negotiate with DM2 the bandwidth reservation within AD2 (this case can be reconducted to the “fork” configuration as depicted in Fig. 2).

It is clear that the outcome of the negotiation between C and VoDpro strongly depends, other than on the actual schedule of VoDpro’s inbound resources, on the outcome of two consecutive negotiations: one between DM1 and DM2 for bandwidth reservation within AD2 and one between VoDpro and DM1 for bandwidth reservation within AD1.

Let us assume that the three negotiations are carried out successfully, i.e., the current availability of resources of VoDpro is sufficient to accommodate C’s request and, furthermore, the amount of network’s resources necessary to support the multimedia streaming can be reserved within both AD1 and AD2.

Right after the negotiation phase, the following SLAs are signed:

- SLA between C and VoDpro for the provision of the multimedia streaming service (SLA1 in Fig. 3);
- SLA between VoDpro and DM1 for the bandwidth reservation within AD1 (SLA2 in Fig. 3);
- SLA between DM1 and DM2 for the bandwidth reservation within AD2 (SLA1 in Fig. 3).

In the described scenario a chain of SLAs have been signed, each representing the QoS level to be guaranteed by each service

![Fig. 3. Service-composition scenario.](image-url)
employed to compose the final service to be delivered to the end consumer.

We draw the attention on the fact that two of the involved actors (namely, VoDpro, e DM), act as both service provider and service consumer. The quality level of the service they provide also depends on the quality level of the service they consume. For instance, the quality of the communication channel between C and VoDpro, provided by DM1 (in this case, acting as a provider) is strongly affected by the quality of the communication channel requested by DM1 (in this case, acting as a consumer) to DM2 within AD2.

During service provision each of the involved actors may raise new needs, to face which it might be necessary to modify the QoS level agreed in the chain of the SLAs. The following sub-scenarios could be envisioned.

Sub-scenario 1. While delivering the multimedia streaming to C, VoDpro receives a new service request from another customer. Because of the high resource utilization, VoDpro is not able to serve the new request. It then decides to re-negotiate with DM1 for a downgrade of the previously agreed QoS level (for instance, a decrease of the frame rate or of the color depth). C agrees, so new resources are now available to VoDpro to accommodate the new service request. As a consequence of the QoS level downgrade, now the bandwidth reserved to C is oversized (the streaming requires less end-to-end bandwidth), hence VoDpro decides to re-negotiate with DM1 in order to request less bandwidth than that previously assigned to itself. DM1 agrees, for instance, it may want to re-assign the gained bandwidth to accommodate new pending requests, and decides on its turn to start a negotiation with DM2 to request a shrink of the bandwidth reserved to itself within AD2. If again DM2 agrees, all the negotiations are successful and the SLAs can be accordingly modified (and the service provision adjusted to the new QoS levels with no interruption of the service itself). We may notice that the adjustment of the quality of the streaming can be done even though DM1 rejected the DM1’s request to shrink the bandwidth reserved in AD2. If again DM2 agrees, DM1 might decide to accept anyway the VoDpro’s request for bandwidth’s shrinking in AD1. Either way, however, the involved actors would get their requests satisfied, and the service provision would not get interrupted.

Sub-scenario 2. The resources of DM1 are not adequate to accommodate the new incoming requests for bandwidth over AD1, DM1 decides then to negotiate a shrinking of the bandwidth previously reserved to VoDpro, which decides to accept provided that C accepts a degradation of the current streaming QoS. In that case, in fact, VoDpro might schedule the just freed resources to serve other pending requests. C chooses to accept the proposal, so the re-negotiations are again successful. Two SLAs can now be modified (the one between C and VoDpro, and the one between C and DM1) and the quality levels of the related services are accordingly adjusted (again, with no service interruption). Like in the previous case, DM2 might want to negotiate with DM1, a shrinking of the bandwidth reserved to it over AD2 to build a connection path to C. If the negotiation was not successful, the SLA previously signed by DM1 and DM2 would still apply to this service provision (In this case, on AD2, there would be an amount of bandwidth reserved and paid by DM1, but not used).

Sub-scenario 3. C is not satisfied with the perceived quality of the streaming and negotiates with VoDpro for an improvement of the parameters associated to the streaming (e.g., the frame rate). VoDpro is able to satisfy the request, but an increase of bandwidth in the path to the user is needed to support the delivery of the augmented streaming. VoDpro negotiates with DM1 to obtain an increase of bandwidth. Provided that DM1 is willing to provide extra bandwidth on its domain, on its turn it has to negotiate with DM2 to obtain more bandwidth on AD2. In this case, the request of C can be satisfied only if both the negotiations for the increase of bandwidth in the two domains are successful. In that case, the three SLAs can be modified and the quality of the services are adjusted to the new agreed levels.

4. The WS-Agreement specification

The WS-Agreement specification (Andrieux et al., 2007) has recently (May 2007) been published as an OGF Proposed Recommendation. The objective of WS-Agreement is to define a language and protocol for:

- establishing agreements between two parties;
- advertising the capabilities and requirements of service consumers and providers;
- creating agreements based on creation offers;
- monitoring the agreement compliance at runtime.

This protocol uses an XML-based language for specifying the nature of an agreement, and agreement templates to facilitate the discovery of compatible agreement parties. It is generally aimed to be a “one-shot” interaction, and is not directly intended to support negotiation. The agreement creation process is restricted to a simple request–response protocol: one party (the Agreement Initiator, generally acting on behalf of a Service Consumer) creates an agreement document, possibly based on an agreement template, and proposes it to the other party (the Agreement Responder, generally acting on behalf of a Service Provider). The responding party evaluates the agreement’s offer and assesses its resource situation before accepting or rejecting the offer.

An agreement between a Service Consumer and a Service Provider specifies one or more service level objectives both as expressions of requirements of the Service Consumer and assurances by the Service Provider on the availability of resources and/or on service qualities. The specification provides a schema for defining the overall structure for an agreement document. The structure of an agreement is illustrated in Fig. 4.

In this structure there are information on the agreement parties and a set of terms. The agreement terms represent contractual obligations and include a description of the service as well as the specific guarantees given. A service description term (element within the Service Term) can be a reference to an existing service, a domain specific description of a service or a set of observable properties of the service. A guarantee term, on the other hand, spec-

![Fig. 4. Structure of an agreement.](image-url)
ifies non-functional characteristics in service level objectives as an expression over properties of the service, an optional qualifying condition under which objectives are to be met, and an associated business value specifying the importance of meeting these objectives.

The conceptual model for the architecture of system interfaces based on WS-Agreement has two layers: the agreement layer and the service layer. The agreement layer implements the communication protocol used to exchange information about SLAs between the Service Customer and the Service Provider (create, represent and monitor agreements), and it is responsible for ensuring that the SLA guarantees are enforced by a suitable service provider. This layer handles well-formed requests to the service layer. The service layer represents the application-specific layer of the service being provided. The interfaces in this layer are domain-specific, and need not be altered when the agreement layer is introduced. The typical WS-Agreement life cycle has four phases:

1. **Exploration**: A Service Provider provides templates describing possible agreement parameters.
2. **Creation**: Consumer fills in parameters, and makes an offer.
3. **Operation**: Agreement state is available for monitoring as a ResourceProperty.
4. **Termination**: Agreement is destroyed explicitly or via soft state (termination time).

The agreement states exposed to an initiator observes the state model depicted in Fig. 5.

The **Observed** state means that an agreement offer has been proposed by the Agreement Initiator and has been accepted by the Agreement Responder; both parties are then obliged with respect to the service and guarantee terms of the agreement. When an agreement offer has been made but it has been neither accepted nor rejected, its state is said to be **Pending**. The **Rejected** state means that an agreement offer has been made and rejected. The **Terminated** state means that an agreement offer has been terminated by the Agreement Initiator and that the obligation no longer exists. Finally, the **Complete** state means that an agreement offer has been received and accepted, and that all activities pertaining to the agreement are finished. For more details about the WS-Agreement specification the reader may refer to (Andrieux et al., 2007).

### 4.1. Limitations

A service level agreement compliant to the current WS-Agreement specification is a rigid contract whose terms can not be modified during its life cycle.

The protocol for the creation of a WS-Agreement is based on a very simple request–response interaction between the parties. In theory there is no upper bound to the response time of a WS-Agreement creation offer, thus the proposal originator, once issued the offer, either waits indefinitely for a response or decide to withdraw it. When a WS-Agreement has been created it can not be modified and is effective until all the activities pertaining to the agreement are finished or until one of the signing party decides to terminate it. The main reason for a Service Provider to terminate an agreement is its inability to provide a service that maintain the QoS levels that were guaranteed in the agreement. Therefore, there is a need for a flexible mechanism enabling the run-time re-negotiation of the guarantees on the QoS once violations on such guarantees are expected to occur. On the other hand, a mechanism that permits the re-negotiations of the guarantees on the QoS would also enable Service Providers and Service Customer respectively to offer and to ask for upgrades of the performance of the service provided at run-time.

The new protocol being designed must take into account the dynamics and the new requirements that the scenarios presented in Section 2 impose. We remark that in such scenarios there are several actors, in the dual role of provider and consumer of services, that stipulate one-to-one agreements with each others. We must consider that:

- a request for the creation of an agreement often triggers several requests of agreement creation in a cascade fashion, giving rise to a “network” of correlated agreements;
- a request of modification of a given agreement may trigger modification requests for the agreements correlated to it;
- the modification to be applied to an agreement must be such as not to justify the cancellation of the agreement and the creation from scratch of a new agreement;
- the party receiving a modification proposal must respond within a useful time that is specified by the proposal originator, otherwise the proposal shall be considered withdrawn;
- the number of modifications that an agreement can undergo at run-time must be agreed by the parties at sign time.

### 5. WS-Agreement extension for dynamic SLAs management

Focusing on the just described requirements, in this work we enhance the flexibility of the WS-Agreement approach by integrating new functionality to the protocol that enable the parties involved in a scenario of service composition to re-negotiate the levels of quality of the services and modify the guarantee terms of the agreements while services are being provided. The proposed protocol does not support the process for the re-negotiation of the QoS levels, but provides the means to adjust the terms of an agreement accordingly to the outcome of an eventual negotiation process. The main modification being introduced to the current specification concerns some integrations to the XML schema of the agreement and an enhancement of the protocol with new interactions for the run-time modification of the agreement.

### 5.1. Modification to the Agreement Schema

We believe that the parts of an agreement that might be modified at run-time, while preserving at the same time the nature of the agreement itself, are those concerning the guarantees on the Service Terms. Specifically, in an agreement document the terms susceptible of modification are the service level objectives (SLOs) in the Guarantee Terms section. In particular, we have introduced a new concept of SLO that can be modifiable according to some rules that limit both the time interval of modifiability of the SLO and the number of modifications that the SLO can undergo. Such
parameters, once agreed by the parties at signing time, can not be further modified during the agreement's life span. Instead, what can be modified at runtime is the objective itself to be achieved.

In the following we report the excerpt of Agreement's XML schema regarding the modifications that have been introduced.

```xml
<xs:element name="ModifiableServiceLevelObjective" type="wsag:ModifiableServiceLevelObjectiveType"/>

<xs:complexType name="ModifiableServiceLevelObjectiveType">
  <xs:sequence>
    <xs:element name="ModificationWindow" type="xs:TimeWindowType"/>
    <xs:element name="MinModificationsTriggerTime" type="xs:PercentageType"/>
    <xs:element name="Slo" type="xs:SloType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SloType">
  <xs:sequence>
    <xs:element name="Objective" type="xs:anyType"/>
    <xs:element name="ScheduledTime" type="wsag:PercentageType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TimeWindowType">
  <xs:sequence>
    <xs:element name="NotEarlierThan" type="PercentageType"/>
    <xs:element name="NotLaterThan" type="PercentageType"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="PercentageType">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0="/>
    <xs:maxInclusive value="100="/>
  </xs:restriction>
</xs:simpleType>
```

First of all, it is worth noticing that an Agreement's XML document conforming to the current specification's Agreement XML schema is conform to the new schema too. The novelties introduced do not prevent Service Providers and Service Customers to stipulate agreements based on the old concept of un-modifiable SLO. At protocol's design time our purpose was, in fact, to maintain the backward compatibility with the original specification. The rest stipulate agreements based on the old concept of un-modifiable SLO.

The modifiable SLO is composed of three elements: ModificationWindow, MinModificationsTriggerTime and Slo. In particular, ModificationWindow represents the time lapse during which carrying out the modification of the SLO is allowed. The upper bound and the lower bound of the interval are expressed as a percentage of the entire life time of the agreement, which spans from the time that the agreement proposal is accepted to the agreement's expiration time (which is specified by the parameter expirationTime in the agreement document). With the MinModificationsTriggerTime element the parties agree on the minimum time lapse between two subsequent modifications to be carried out on a given SLO. It has a percentage type, and is to be meant as a fraction of the ModificationWindow interval. This value, as showed later, fixes the maximum number of modifications that can be carried out on a SLO.

The Slo term represents the scheduling of a modification of the service level objective. It is composed of the Objective element, which contains the specification of the new service level objective, and of the Scheduled element, of percentage type, from which it is possible to come to the scheduling time of the modification. The Slo term's maximum occurrence is unbounded. The first occurrence of this element, actually, represents the service level objective that the parties had agreed upon when the agreement was signed. All the subsequent occurrences represent the scheduling of modifications on the service level objective agreed, from time to time, by the parties. Obviously, there can be no more occurrences than the maximum number of allowed modifications.

Let then $T_a$ be the agreement's life span. The instants of time $T_{ne}$ and $T_{nl}$ that bound time interval within which modifications can be carried out are defined as follows:

\[
T_{ne} = \text{NotEarlierThan} * T_a \\
T_{nl} = \text{NotLaterThan} * T_a
\]

Through the Scheduled element it is possible to come to the modification scheduling time:

\[
T_{mod} = T_{ne} + \text{Scheduled} \times (T_{nl} - T_{ne})
\]

The minimum time lapse between two subsequent modifications on a given SLO can be calculated in the following way:

\[
T_{slot} = \text{MinModificationsTriggerTime} \times (T_{nl} - T_{ne})
\]

From $T_{slot}$ we can derive the maximum number of allowed modifications:

\[
\text{MaxCount} = 1 + (T_{nl} - T_{ne})/T_{slot}
\]

Fig. 6 shows the timing for the modification process within the entire agreement's life span. In this specific case, we have depicted two modifications that have been agreed at run-time by the parties, and have been scheduled at $T_{mod1}$ and $T_{mod2}$ respectively, being of course $T_{mod1} - T_{mod2} < T_{slot}$.

5.2. Enhancement of the Protocol

We have so far established the requirements and the criteria for the modification of an agreement. Now we describe the means available to the parties of an agreement to handle its modification.

The current WS-Agreement protocol provides only one single interaction. The Agreement Initiator (AI from now on) makes an agreement offer and the Agreement Responder (AR from now on) is called to accept or reject it. We introduce a new form of interaction that enables both the parties to request modifications of the guarantees of an earlier signed agreement. Either the AI or the AR, therefore, according to the above mentioned criteria, can make offers to modify one or more Agreement's SLOs; such modification becomes effective only after the other party has accepted the proposal.
We stress that after that an agreement's modification offer has been issued, the service provisioning is not interrupted, nor the agreement's monitoring is suspended. If the responding party accepts the proposal, the Service Provider will have to adjust the QoS to the new agreed levels; if the proposal is rejected, the service provisioning will continue according to the original QoS levels. The rejection of an agreement's modification offer, in fact, does not invalidate the agreement itself. The same applies to eventual multiple modification offers: if the \((i + 1)\)th modification proposal is rejected, the guarantees agreed in the context of the \(i\)th proposal still apply.

In the following we detail the new Port Types and Operations that have been introduced for the integration of the described functionality.

First of all, in order to be able to accept modification offers, each party must provide the other with an ad-hoc contact point (or End Point Reference – EPR). We have introduced a new operation for the creation of Modifiable Agreements. Such operations differ from the ones provided in the current WS-Agreement specification (CreateAgreement) for the fact that the two parties exchange each other's EPRs that are specific for receiving agreement's modification proposals.

We thus have added the CreateModifiableAgreement operation to the “AgreementFactory” Port Type. Its signature is reported below.

\[
\text{wsag: CreateModifiableAgreement}
\]

\[
\begin{align*}
\text{Input} & \quad \text{Result} \\
\langle \text{wsag: CreateModifiableAgreementInput} \rangle & \quad \langle \text{wsag: CreateModifiableAgreementResponse} \rangle \\
\langle \text{wsag: InitiatorAgreementEPR} \rangle & \quad \langle \text{wsag: CreatedAgreementEPR} \rangle \\
\langle \text{wsag: EndpointReference} \rangle & \quad \langle \text{wsag: CreatedAgreementEPR} \rangle \\
\langle \text{wsag: InitiatorAgreementEPR} \rangle & \quad \langle \text{wsag: ResponderAgreementEPR} \rangle \\
\langle \text{wsag: EndpointReference} \rangle & \quad \langle \text{wsag: ResponderAgreementEPR} \rangle \\
\langle \text{wsag: InitiatorAgreementProposal} \rangle & \quad \langle \text{wsag: CreatedModifiableAgreementResponse} \rangle \\
\langle \text{wsag: AgreementOffer} \rangle & \quad \langle \text{wsag: CreatedModifiableAgreementResponse} \rangle \\
\langle \text{wsag: CreateModifiableAgreementInput} \rangle & \quad \langle \text{wsag: CreatedModifiableAgreementResponse} \rangle
\end{align*}
\]

Similarly, we added the CreateModifiablePendingAgreement operation to the “PendingAgreementFactory” Port Type. Its signature differs from that of the CreateModifiableAgreement operation for just one input parameter. Since it is not relevant to our work, we omitted to report its details.

Let us focus back to a service composition scenario, like one of those depicted in Figs. 1 and 2. Let us suppose that the AI acting on behalf of the customer C has made his offer of agreement’s creation to the AR, which acts on behalf of SP, by invoking one of the two operations described above. In such a scenario this proposal triggers a number of consecutive proposals in a cascade fashion. If just one of this proposal got rejected, the original AI’s proposal would be rejected. Let us suppose that all the agreement’s offers have been accepted, so that the AI gets its agreement proposal accepted as well. From this moment on, each of the parties of this scenario is entitled to make a proposal of modification of the agreements that it is involved in.

In the general case, for the provision of a given service a party signs as many agreements as needed to provide that service. For the provision of a given service, a party signs as many agreements as needed to provide that service. When that party for any reason wants to modify one of these agreements, it first has to ask itself what the consequences of that modification might be, i.e., what that modification might mean for the rest of the agreements in that scenario. After the modification of a given agreement, in fact, the party might not be able to honor the rest of the agreements signed with its neighbors, and the neighbors, in their turn, might not be able to honor the agreements signed with their own neighbors, and so on.

To make it clearer, let us refer to the tree structure depicted in Fig. 2. Suppose that SP wants to modify the agreement signed with SP. It has first to verify whether, because of such a modification, it is no more able to honor both (or even one of) the agreements, respectively, signed with SP and SP. If SP is not able to do that, then it has two to choose between either giving up to request the modification of the agreement signed with SP or checking whether SP and SP are willing in their turn to modify the agreements that it would not be able to honor. Recursively, when receiving SP’s proposal of agreement modification, before accepting such proposal, SP has to check whether, in the case that it decided to accept, it would still be able to honor the agreement signed with SP. If not, it might test SP’s willingness to modify that agreement, and so on.

This is to say that a proposal of agreement modification, whenever it is originated, can propagate throughout the agreement tree, theoretically affecting every agreement. The acceptance of a given modification proposal is bound to the acceptance of other agreements’ modification proposals in the tree structure.

To support the dynamic modification of agreements in the WS-Agreement specification, we introduce the “AgreementHandler” Port Type. This Port Type must be implemented by every party interested in taking part in the management of dynamic agreements. Two operations are exposed through this Port Type: ModifySLO and CondModifySLO. The ModifySLO operation is defined in the following.

The result parameter is just an acknowledgment for the reception of the modification offer; it does not represent the response to the offer itself. The input parameter SLOModificationOffer represents the offer of an agreement where modifications have been introduced to any of the modifiable SLO. The input parameter RequestExpirationTime represents the time-to-live for this specific request. The input parameter SLOModificationAcceptanceEPR represents the contact point of the party requesting the modification,
to which the responding party will have to notify its decision of acceptance or rejection of the SLO modification proposal. To support such call-back mechanism, a new Port Type has been introduced. It has been named "SLOModificationAcceptance", must be implemented by both the parties and is in charge of receiving notifications for SLO modification proposals' acceptance or rejection. Two operations are exposed by this Port Type, respectively Accept and Reject. Here follows the signature of Accept:

By the time that the Accept operation is invoked, the modification of the SLO is effective, i.e., the agreement has been modified and the Service Provider must adequate the QoS level to the new guarantee. As for the Reject operation, its signature is very similar to that of Accept, thus we omitted to report it.

For a better comprehension of the ModifySLO operation, in Fig. 7 we report a very simple sequence diagram referred to the scenario depicted in Fig. 1. The Service Provider SP3 (or the AI on behalf of it) propose to SP2 (or the AR on behalf of it) a modification of a SLO term of the Agreement SLA32. This proposal triggers other modification offers in the chain. In this case all the triggered offers are accepted, so the original proposal is accepted as well. If just one of the involved actors rejected a modification offer then the original proposal would not be accepted.

Let us now assume SP1 be the party needing to modify the terms of one of its agreements. This time the party originating the modification proposal has two agreements with two different parties (two edges linking to two nodes). If for instance, according to the SP1's needs, the modification of a given SLO in the Agreement SLA21 would cause the need of a modification of a SLO in the Agreement SLA12, SP1 must check that both C and SP2 agree with those modifications. Since the acceptance of one modification proposal is bound to the acceptance of the other, the ModifySLO operation is not appropriate to handle this situation. We need a request–response-commit interaction in order to first gather the willingness of both C and SP2 to accept the proposed modifications, and then to confirm them (i.e., to commit the agreements' modifications). To this end, to the "AgreementHandler" Port Type we have added the ConditionalModifySLO operation, whose input/result parameters' definition does not differ from those of the ModifySLO operation. The parties receiving a proposal of conditional modification, can respond by calling either a ConditionalAccept or a ConditionalReject (both of which have been added to the "SLOModificationAcceptance" Port Type) on the modification originator. The signature of ConditionalAccept is similar to that of Accept, but its meaning is different: the party invoking a ConditionalAccept communicates that it is just willing to accept the modification proposal, but a confirmation is still needed in or-

5.3. Implementation work and use cases

This section will give an insight on the implementation work for the creation of a real framework prototype for the management of dynamic SLAs. The implemented framework has been then used within a composite service scenario, namely the one presented in Section 3.

The prototype has not been implemented from scratch, but is based on CREMONA (Dan et al., 2004), a framework providing tools for the management of SLAs. More precisely, our work is grounded
on the ETTK environment from IBM, which provides libraries (CREMONA) and web tools to manage SLAs according to the WS-Agreement specification. What has been done so far can be summarized in the following:

- enrichment of the CREMONA library with new API for the creation and management of modifiable SLAs;
- set-up of a B2B scenario, involving a composition of services;
- testing of two use cases for the previous scenario, demonstrating the benefits of the proposed approach.

In the class diagram of Fig. 9 we outline the new port types and operations that enrich the CREMONA API with the new features. New Web services and Java components, implementing the new API features previously described, have been developed and added to the ETTK framework.

In order to test the viability of both the new approach and the new API, the service composition scenario described in the Section 3 has been set up, and two use cases have been implemented. Here we present the sequence diagrams related to those use cases. In Fig. 10 the Customer C requests the modification of the agreement signed with VoDprov. The requests propagates through the chain of the Service Providers involved in the provision of the service to C. The agreement modification is committed only if all the requests are accepted, as it is shown in the sequence diagram. As a result of this process new agreements are established among the several parties involved in the scenario. We also tested the case where one of the domain providers (DM1 or DM2) is not able to increase its offered bandwidth and so is not willing to modify the signed SLA: in this case the whole modification process fails and the old agreements continue to be effective.

In Fig. 11 DM1 proposes a reduction of the bandwidth to VoDprov, which decides to accept provided that C accepts a degradation of the current quality of the streaming. Afterwards, DM1 proposes to DM2 a reduction of the bandwidth previously assigned to VoDprov in AD2.
6. Related works

Among the standardization efforts taken to define a common and shared framework for the definition and the monitoring of service level agreements, two are worth citing.

WSLA (Keller and Ludwig, 2003) is a framework developed by IBM for specifying and monitoring service level agreements for Web Services. WSLA provides a formal language based on XML Schema to express SLAs and a runtime architecture which is able to interpret this language. The framework is able to measure and monitor the QoS parameters of a Web Service and reports violations to the parties of the SLA.

The Web Services Agreement Specification (WS-Agreement) (Andrieux et al., 2007) from the Open Grid Forum (OGF) describes a protocol for establishing an agreement on the usage of services between a service provider and a service consumer. This protocol uses an XML-based language for specifying the nature of an agreement, and agreement templates to facilitate the discovery of compatible agreement parties. For monitoring purpose, the agreement status is exposed to the parties, so they can verify anytime whether the agreement is violated or fulfilled. The protocol is generally aimed to be a "one-shot" interaction (request–response). Currently it does not support any negotiation mechanism. WS-Agreement is now a proposed OGF recommendation for the expression and creation of SLAs. Recently the standardization efforts of the OGF are focusing on the integration of new features to support the negotiation and the re-negotiation process of the SLAs. Several issues presented in this paper are being discussed by the GRAAP working group; the main concern now is the design of a robust signalling protocol for the negotiation of the QoS in a distributed environment.

CREMONA (Dan et al., 2004) is a WS-Agreement framework implemented by IBM. Cremona (Creation and Monitoring of Agreements) proposes an architecture for the WS-Agreement-imple-
menting middleware. In addition, the Cremona Java Library implements the WS-Agreement interfaces, provides management functionality for agreement templates and instances, and defines abstractions of service-providing systems that can be implemented in a domain-specific environment.

WS-Agreement Framework (WSAG4J) (WS-Agreement framework, 2007) provides means to implement WS-Agreement services and the respective client-side access. It is middleware-agnostic, fully compliant with the WS-Agreement Specification version 1.0 and implemented in Java. Currently some WS-Agreement features (PendingAgreement and AgreementAcceptance) are not supported.

In GRID and SOA environments SLAs have been widely employed to establish agreements on the quality level of non-functional parameters that a service provider must grant whilst delivering a service to a customer. Many works in literature have embraced the SLA concept and have proposed frameworks for “SLA-based” resource management and scheduling. Within such frameworks SLAs have proved to be of great support for dealing with issues related to many services such as service discovery, advanced resource reservation, resource scheduling, service and resource monitoring.

In the VIOLA (Waeldrich and Ziegler, 2006) project a resource reservation mechanism based on WS-Agreement is responsible for the negotiation of resource allocation with the local scheduling systems. A Three-phase-commit-protocol (3PCP) is proposed to overcome the limitations of WS-Agreement. It is based on the creation of different types of agreements within a negotiation process, namely a Declaration of Intention Agreement, a Preparation
Agreement, and a Commitment Agreement. All of these agreements are normal WS-Agreements, following a certain naming convention. This protocol basically aims at solving problems related to the creation of agreements on multiple sites. The AssesGrid (Battre et al., 2007) project focuses on risk management and assessment in Grid computing. The project's aim is to provide the service providers with risk assessment tools that will help them to make decisions on suitable SLA offers, by relating the risk of failure to penalty fees. Similarly, end-users get knowledge about the risk of an SLA violation by a resource provider that helps to make appropriate decisions regarding acceptable costs and penalty fees. A broker is the matchmaking between end-users and providers (i.e., the contractors). A negotiator module is responsible for negotiating SLAs with external contractors using the WS-Agreement-Negotiation protocol, which is an early version of the WS-Agreement protocol including little support for the negotiation.

GREEIA is a service-oriented infrastructure designed to support grid collaborations through service provision across organizational boundaries. The framework includes a service manager with the ability to define the available resources (e.g., CPUs and applications), assign portions of the resources to users by way of service-level agreements (SLAs) and bill for resource usage. Furthermore, a monitoring service is responsible for monitoring the activity of the services with respect to the agreed SLA.

The BREIN consortium is working on a business framework prototype that will foster electronic business collaborations by taking the classical Grid approach. Among the others, the prototype's claimed capabilities are: Service Discovery with respect to SLA capabilities, SLA negotiation in a single-round phase, system monitoring and evaluation, SLA evaluation with respect to the agreed SLA. The WS-Agreement/WSLA specifications will be used to support the SLAs management. Dynamic SLAs are in the focus of the project.

In the context of the BEingGrid project, eighteen Business Experiments, designed to implement and deploy Grid solutions in industrial key sectors, have been conducted on top of a middleware of Grid services from across the Grid research domain. Almost half of the Experiments need a strong SLA support, thus justifying the need for a SLA management tool within a Business Grid. In the future a study on the composition of SLAs across business domains will be undertaken. This initiative witnesses the interest that the industry is demonstrating in the issues concerning SLAs management. Dynamic SLAs are in the focus of the project.

In Joita et al. (2005) the WS-Agreement specification is used as a basis on which negotiations between two parties may be conducted. An agent-based infrastructure takes care of the agreement offer made by the requesting party; from this offer many one-to-one negotiations are started in order to find the service that best matches the offer.

The discussion about the opportunity of having dynamically modifiable SLAs is still open. We believe that, in order to satisfy the business requirements coming from GRID and, more generally, SOA environments this opportunity should be taken. The OGF as well as many European Grid projects (Battre et al., 2007; GRIA; BREIN; BEingGRID) are investigating on dynamic SLAs, but much work still has to be done in order to clearly identify the business requirements behind them.

In Frankova et al. (2006) the authors propose an extension of WS-Agreement allowing a run-time re-negotiation of the guarantees. Some modifications are proposed in the section wsag:GuaranteeTerm of the agreement schema, and a new section is added to define possible negotiations, to be agreed by the parties before that the Agreement offer is submitted. In this work there is not a real real-time re-negotiation because, after the agreement's acceptance, there is no interaction between the Service Provider and the Service Consumer.

In Sakellariou and Yarmolenko (2005) the authors specify the guarantee terms of an agreement as functions rather than just values. The aim of the work was to minimize the number of re-negotiations necessary to reach some consensus on values associated with agreement terms.

In Pichot et al. (2007) the authors present a bilateral WS-Agreement based negotiation protocol used to dynamically negotiate SLA templates. They propose a simple extension of the WS-Agreement protocol in order to support a simple offer/counter-offer mode.

To our knowledge, in literature there is little effort towards the definition of the requirements that service composition scenarios impose on the design of an infrastructure for the creation and management of SLAs. Our contribution does not aim at giving an answer to the (yet unclear) questions arising in the complex scenario that we have depicted. We have though showed the need (both for the provider and the customer) for an improved flexibility of WS-Agreement protocol, allowing for run-time modification of some guarantee terms of a signed agreements. The proposal maintains backward compatibility with existing specification, so that providers and customers are free to choose the best option for their environment (Di Modica et al., 2007a). Negotiation protocols have been kept out of the proposal since we believe they should not be included within any protocol for agreement management: providers and customers should be able to negotiate QoS levels by using whichever negotiation schema that best suits their needs.

7. Conclusion

In this work some B2B scenarios have been analyzed in order to identify the requirements that a framework for the managements of one-to-one SLAs must fulfill. Starting from the WS-Agreement specification we have proposed the integration of new functionality to improve the flexibility of the management of SLAs in service composition scenarios. We have then implemented the resulting protocol within a real framework for the creation, monitoring and modification of WS-Agreements, and built upon it a test-bed for evaluating the effectiveness and the performance of the proposed protocol. In the future we are planning to further improve the flexibility of an agreement, also accounting for the possibility of modifying at run-time the agreement's expiration time. Another objective for our future work is the identification of the features that a service must exhibit in order for a re-negotiation of its QoS guarantees to be feasible at run-time, without requiring the suspension of the service provisioning.

References


Giuseppe Di Modica received his Degree in Computer Engineering in 2000 from the Engineering Faculty of University of Catania (Italy). Since then he has been engaged in research on distributed systems with the Department of Computer and Telecommunications Engineering of Catania University. In 2005 he received the Ph.D. in Computer Science and Telecommunication Engineering from University of Catania. The Ph.D. thesis discussed was entitled “A user-centric analysis of the interworking between heterogeneous wireless systems”.

His research interests include mobile agents, ad-hoc networks, and wireless networking, mobile P2P computing, Grid and service oriented architectures. He is currently a research assistant at Engineering Faculty of University of Catania (Italy).

Orazio Tomarchio received his Degree in Computer Engineering from the Engineering Faculty of University of Catania (Italy) in 1995. Since then he has been engaged in research on distributed systems with the Department of Computer and Telecommunications Engineering of Catania University.

In 1999 he received the Ph.D. in Computer Science from University of Palermo. The Ph.D. thesis discussed was entitled “Design and implementation of a mobile agent platform and its application for distributed systems management”.

The scientific activity of Dr. Tomarchio has been focused on studying distributed systems, particularly with regard to programming and management techniques. His main research interests include mobile agents, network and QoS management techniques, mobile P2P computing, middleware for MANETs, Grid and service oriented architectures.

He is currently an assistant professor at Engineering Faculty of University of Catania (Italy).

Lorenzo Vita received the electrical engineering degree from the University of Catania in 1979.

Since 1980 he has been cooperating in many research fields with the Institute of Computer Science and Telecommunications of Catania University, where he was Associate Professor.

Initial scientific activity was focused on parallel processing, distributed and parallel systems, multimedia systems and network management. Recently, his research interests also include mobile agents, Grid and service oriented architectures.

He is currently Full Professor of Software Engineering at the University of Catania, Italy.