# Semantics and Extensions of WS-Agreement

Ganna Frankova, Daniela Malfatti, and Marco Aiello
Dept. of Information and Communication Technologies, Univ. of Trento, Trento, Italy
Email: ganna.frankova@unitn.it
daniela.malfatti@studenti.unitn.it
marco.aiello@unitn.it

*Abstract*— **When having repeated interactions with a service provider, a service consumer might desire guarantees on the delivery of the service. These guarantees involve both functional and non-functional properties of the offered service over a number of invocations. When the guarantee terms are explicitly defined in a document, we talk about a service level agreement. WS-Agreement is an industry driven emerging protocol for the specification of agreements in the context of Web Services. If, on the one hand, WS-Agreement defines the XML syntax for the language and protocol, on the other hand, it gives only a vague textual overview of the intended meaning. We fill this gap by providing a formal definition of an agreement and analyzing the possible evolutions of agreements and their terms over an execution. As a result we identify a number of extensions which involve the initial negotiation, the monitoring of running agreements, and the possibility of renegotiating agreements in executions. We evaluate the proposed approach through experimentation.**

*Index Terms*— **Service-Oriented Computing, Service Level Agreement, Quality of Service, Web Services.**

## I. Introduction

The construction of massively distributed and loosely coupled applications is becoming evermore a reality thanks to the introduction of Web Services (WS). WS are a family of XML-based protocols which cover the issues of describing, publishing, and finding individual services, as well as describing messaging and coordination mechanisms, quality of service parameters and many more facets tied to the realization of widely distributed information systems. One of the key issues in Web Services is that of automatically composing individual operations of services in order to build complex added-value services. The research on composition is well under way, but most of the focus is on functional properties of the composition, that is, how does one automatically compose? How does one enrich the services with semantic self-describing information? How does one discover the available services to use for the composition? If, on the one hand, this is crucial, on the other one, it is not enough. Non-functional properties of the composition are also of paramount importance in defining the usability and success of a composed service. Think for instance of desiring a service that performs a biological computation composing the services offered by a number of web service enabled machines. If the user knows that the composition is correct with respect to his goal, they will be satisfied with the answer they receive, but if the answer takes 3 years to be delivered to the user, the correctness is of little use. Therefore, the quality of a composed service is very important when interacting with an asynchronous system built out of independent components.

With the term Quality of Service (QoS) we refer to the non-functional properties of an individual service, or a composition of services. The term is widely used in the field of networking. Usually it refers to the properties of availability and performance. In the field of Web Services, the term has a wider meaning. Any non-functional property which affects the definition and execution of a web service falls into the category of QoS, most notably, accessibility, integrity, reliability, regulatory, and security [1]. Dealing with QoS requires the study of a number of problems. One, the design of quality aware systems. Two, the provision of quality of service information at the level of the individual service. Three, ensuring that a promised quality of service is actually provided during execution. In [2], we addressed the first issue by using the Tropos design methodology, and the second one by resorting to WS-Policy to describe QoS properties. In this paper, we consider the second and third issues; in particular, we show how to provide a framework to negotiate the provision of a service according to a predefined QoS, and how to handle changes during the interactions of Web Services, and how to prevent the QoS conditions failure.

WS-Agreement is an extensible markup based language and protocol for advertising the capabilities of providers, creating agreements based on initial offers, and for monitoring agreement compliance at run-time. The motivations for the design of WS-Agreement stem out of QoS concerns, especially in the context of load balancing heavy loads on a grid of web service enabled hosts [3]. However, the definition of the protocol is totally general and allows for the negotiation of QoS in any web service enabled distributed system. If, on the one hand, the proposal of WS-Agreement is a step

Based on "What's in an agreement? An analysis and an extension of WS-Agreement," by M. Aiello, G. Frankova, and D. Malfatti which appeared in the Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC 2005), Amsterdam, The Netherlands, December 2005. © 2005 Springer

© 2006 ACADEMY PUBLISHER

forward for obtaining web service based systems with QoS guarantees, on the other hand, the protocol proposal is preliminary. The current specification [4] defines XML syntax for the language and protocol, and it gives a vague textual overview of the intended semantics, without defining a set of formal mathematical rules. Furthermore, a reference architecture is proposed to show how WS-Agreement are to be handled, [5]. Nevertheless, a formal analysis of what an agreement is still missing.

We address the question *What's in an Agreement?*. In particular, we provide a formal analysis of WS-Agreement by resorting to finite state automata, we provide a set of formal rules that tie together agreement terms and the life-cycle of an agreement. From the analysis, some shortcomings of the protocol become evident. Most notably, the protocol does not contemplate the negotiation of the agreement itself, furthermore, there is no checking of how close a term is to being violated and, even more, breaking one single term of the agreement results in terminating the whole agreement, while a more graceful degradation is desirable. To overcome these shortcomings, we propose an extension of WS-Agreement for which we provide appropriate semantics, that allows (i) early warnings before agreement violation, and (ii) negotiation and possibly renegotiation of running agreements.

The remainder of the paper is organized as follows. Related work is discussed in Section II. In Section III, we present the WS-Agreement protocol defined in [4]. In Section IV, we propose a formal definition of an agreement and of its life-cycle. Section V is devoted to the presentation of an extension of WS-Agreement with the goal of defining negotiation and improving the duration of an agreement in execution. Experimental results are given in Section VI. Conclusions and open issues are presented in Section VII.

## II. Related Work

QoS issues in the context of Web Services is gaining increasing attention in the Service-Oriented Community. On the one hand, research focuses on QoS metrics and models, quality-driven web service discovery and selection, and on QoS aggregation for web service composition. On the other hand, there are several approaches to SLA specification, negotiation and monitoring. Next, we overview QoS metrics, models, and SLA related approaches.

### A. QoS Metrics and Models

Various approaches for modelling quality of service for web services exist. Ran [6] organizes the aspects of QoS into categories, i.e., runtime, transaction support, configuration management and cost, security. The author argues that each category needs to have a set of quantifiable parameters or measurements. An approach for defining QoS requirements is QML [7]: a language for QoS description using XML. QoS aspects are qualified by characteristics as direction and value type. A set of measures for reliability and performance are proposed. Atzeni and Lioy [8] overview security system assessment methods and metrics. WS-Policy [9] defines a framework and model for expressing capabilities, requirements, and general characteristics of individual services. In [10], it is argued that networking issues have need to be taken into account by both web service providers and consumers.

A number of approaches to QoS description of services rely on extensions of the Web Service Description Language (WSDL), e.g., [11], [12]. The main idea is simple: provide syntax to define terms which refer to non-functional properties of operations. Given such description, one can then build a framework for the dynamic selection of Web Services based on QoS requirements. On the negative side the QoS definition is tied to the individual operation, rather to the service as a whole. Furthermore, there is no run-time support, i.e., once a quality parameter is set, it can not be changed at execution time. In [13], [14], the description of elementary service qualities as a quality vector each component of which is a quality parameter for the service is proposed. In [15] Lin, Xie, Guo and Wang propose a fuzzy way to express QoS requirements. According to the fact that some QoS metrics such as response time and invocation price can be changed at run-time, the approaches dealing with rigidly fixed values is not appropriate. Adding a new data structure to the UDDI model in order to take into account non-functional properties is presented in [6]. As the description of quality of service information is static, i.e., it is specified for a particular service and can not be changed at run-time: the approach does not allow to cope with the problem of run-time support. Furthermore, although the approach allows service discovery, it does not support a quality-driven web service composition. As users rate services based on their expectations on the quality of service and the expectations are often different, in [16] the authors propose a quality of service management framework based on users' expectations. However the idea is feasible, the works is at an early stage and there is a need to consider how the expectations can be presented and matched. Maximilien and Singh [17] develop an ontology-based framework for dynamic QoS-aware Web Services selection. If on the one hand, the approach takes into account provider's policies and consumer preferences, on the other hand, negotiation is not allowed. In addition, a semantic web approach, in which services are searched on the basis of the quality of semantically tagged service attributes is presented in [18]. The use of the agent-oriented methodology Tropos to model a wide spectrum of quality of Web Services properties is proposed in [2]. The feasibility of using constraint programming to improve the automation of web services procurement is shown in [19].

### B. SLA Specification, Negotiation and Monitoring

Modelling and measuring QoS is only one aspect of the management and procurement of services. The other half

© 2006 ACADEMY PUBLISHER

of the picture is the negotiation of QoS aspects between services that may have recurring interactions. When the latter occurs, service producers and consumers may need to establish agreements on service provisioning which involve the non-functional properties of the services. Key factors of involved in service level agreements are the specification, negotiation and monitoring of SLAs [20], [21].

Several languages for specifying SLAs have been proposed, most notably, IBM's Web Service Level Agreement (WSLA) Language [22] focuses on web service interactions. The goal of WSLA is twofold: at deployment time it helps the interacting parties to configure their resources to meet a predefined SLA; at run time it helps the interacting parties to monitor the performance of each other and to detect and notify violations. However, the framework does not answer the question "how close a guarantee is to being violated." SLAng [23] is an XML-based language that describes QoS properties to include in SLAs. SLAng does not focus only on web service interactions, but also to specify SLAs for hosting service provisioning (between container and component providers), communication service provisioning (between container and network service providers) and so on. Although SLAng is expressive enough to represent the QoS parameters included in SLA, more work is needed on the definition of the semantics of SLAng. Web Service Offering Language (WSOL) [24] focuses on web service interactions. The language is used to formally specify various constraints, management statements, and classes of services for Web Services. A method to convert the contract from text into an electronic equivalent that can be executed and enforced is presented in [25]. The authors propose using finite state machines to describe standardized conventional contracts.

The negotiating of service agreements has a vital role in the life-cycle of a service level agreement. Presently, negotiation is mainly a manual process and full or partial automation is needed. The term *negotiation* refers to the comprising exchanges of messages, such as offers and acceptance messages between two or more entities [26], [27]. Theoretical bases of SLA negotiation are provided by Demirkan, Goul and Soper [28]. The authors identify negotiation support system requirements. The critical issue is a common understanding of the terms among negotiating parties, i.e., there is an ontology problem of electronic negotiations [29]. Using templates is a proposed solution [30]. Gimpel at al. [26] propose PANDA - Policy-driven Automated Negotiations Decision-making Approach. The approach automates decision-making within negotiation.

Monitoring an established SLA is essential for a service consumer. Non-functional monitoring is concerned with the statistical QoS metrics collection to evaluate wheatear a provider complies with the QoS level specified in the SLA [31], [21]. Fundamental concepts of non-functional SLA monitoring are presented in [31] which contains a discussions on the separation of the computation and communication infrastructure of the provider, service points of presence and metric collection approaches. The authors propose an architecture for QoS monitoring by third parties to ensure that the results are trusted by both the provider and consumer. A Web Service Level Agreement framework for defining and monitoring SLAs is presented in [32]. The work addresses the definition of a language for SLAs specification, creation, and the implementation of a SLA compliant monitor.

A recent proposal is the Web Services Agreement Specification [4], which is described in the next section. In [33], the Agreement-Based Open Grid Service Management (OGSI-A) model is proposed. Its aim is to integrate Grid technologies with Web Service mechanisms and to manage dynamically negotiable applications and services, using WS-Agreement. The WS-Agreement is supported by the definition of a managing architecture: CREMONA - An Architecture and Library for Creation and Monitoring of WS-Agreement [5]. The Web Services Agreement Specification defines the interaction between a service provider and a consumer, and a protocol for creating an agreement using agreement templates. The above approaches show that frameworks for QoS definition and management are essential to the success of the web service technology, but there are a number of shortcomings that still need to be addressed. First, a formal definition of the semantics of a QoS negotiation is missing. Second, the frameworks should be more flexible at execution time because actual qualities of services may change over time during execution.

## III. WS-AGREEMENT

In order to be successful, web service providers have to offer and meet guarantees related to the services they develop. Taking into account that a guarantee depends on actual resource usage, the service consumer must request state-dependent guarantees from the service provider. Additionally, the guarantees on service quality must be monitored and service consumers must be notified in case of failure of meeting the guarantees. An agreement between a service consumer and a service provider specifies the associated guarantees. The agreement can be formally specified using the WS-Agreement Specification [4].

A WS-Agreement is an XML-based document containing descriptions of the functional and non-functional properties of a service oriented application. It consists of two main components that are the agreement Context and the agreement Terms. The agreement Context includes the description of the parties involved in the agreement process, and various metadata about the agreement. One of the most relevant components is the duration of the agreement, that is, the time interval during which the agreement is valid.

Functional and non-functional requirements are specified in the Terms section that is divided into Service Description Terms (SDTs) and Guarantee Terms. The first provides information to define the services functionalities

© 2006 ACADEMY PUBLISHER

that will be delivered under the agreement. An agreement may contain any number of SDTs. An agreement can refer to multiple components of functionalities within one service, and can refer to several services. Guarantee Terms define an assurance on service quality associated with the service described by the Service Description Terms. An agreement may contain zero or more Guarantee Terms.

In [34] a definition for guarantee terms in WS-Agreement is specified and a mechanisms for defining guarantees is provided. An agreement creation process starts when an agreement initiator sends an agreement template to the consumer. The structure of the template is the same as that of an agreement, but an agreement template may also contain a Creation Constraint section, i.e., a section with constraints on possible values of terms for creating an agreement. In [35] enabling of customizations of terms and attributes for the agreement creation is proposed. After the consumer completes in the template, they send it to the initiator as an offer. The initiator decides to accept or reject the offer depending on the availability of resource, the service cost, and other requirements monitored by the service provider. The reply of the initiator is a confirmation or a rejection.

An agreement life-cycle includes the negotiation, implementation, termination and monitoring of agreement states. Fig. 1 shows a representation of the life-cycle. When an agreement is implemented, it does not imply that it is monitored. It remains in the `not_observed` state until services start their execution. The semantics of the states is as follows: `not_observed`: the agreement is implemented and is in execution, but no service involved in the agreement is running; `observed`: at least one service of the agreement is running; and `finished`: the agreement terminates either successfully or not.

## IV. What's in an Agreement?

The WS-Agreement specification provides XML syntax and a textual explanation of what the various XML tags mean and how they should be interpreted. Thank to the syntax, it is possible to prepare machine readable agreements, but a formal notion of agreement is missing. In this section, we formalize the notion of agreement by defining its main components.

*Definition 1 (Term):* A *term* $t$ is a couple $(s, g)$ with $s \in S$ and $g \in G$, where $S$ is a set of $n$ services and $G$ is a set of $m$ guarantees. $T \subseteq S \times G$ is the set of the terms $t$.

In words, a term involves the relationship between a service $s$ and a guarantee $g$, not simply a specific tag of the agreement structure. If the service $s$ appears in the list of services, which the guarantee $g$ is applied to, it means that the couple $(s, g)$ is a term. The number of terms varies between 0 and $n \cdot m$, where 0 means that there is no association between services and guarantees, and $n \cdot m$ indicates the case where each guarantee is associated with all services.

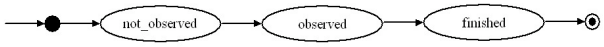


Fig. 1. The life-cycle of a WS-Agreement.

*Definition 2 (Agreement):* An *agreement* $A$ is a tuple $\langle S, G, T \rangle$, where $S$ is a set of $n$ services, $G$ is a set of $m$ guarantees, and $T$ is the set of the terms $t$.

In the following analysis, it is more convenient to consider the agreement as a set of *Terms* rather than a set of related services and guarantees. From the definition of WS-Agreement, we say that an agreement can be in one and only one of three states: `not_observed`, `observed` and `finished`.

*Definition 3 (External state):* The *external state* $A_{es}$ of an agreement $A$ is an element of the set {`not_observed`, `observed` or `finished`}.

We call the above state external, as it is the observable one. We also define an internal state of an agreement, which captures the state of the individual terms.

*Definition 4 (Internal state):* The *internal state* $A_{is}$ of an agreement $A$ is a sequence of terms' states $ts_1, \ldots, ts_p$ of maximum size $n \cdot m$, where $ts_i = (ss_j, gs_k)$ represents the state of $g_k$ guarantee with respect to the state of the $s_j$ service. Service and guarantee states range over the following sets, respectively:
$ss_j \in$ {`not_ready`, `ready`, `running`, `finished`}, and
$gs_k \in$ {`not_determined`, `fulfilled`, `violated`}.

From the definition of *Term*, we see that services and guarantees are related and we can define the internal state of an agreement, but it is necessary to distinguish between terms that have the same service and terms that have the same guarantee.

Proceeding in our goal of answering the question of what is in an agreement, we define the relationship between the internal and external state of an agreement $A$. First, we note that not all state combinations make sense. For instance, it has no meaning to say that a guarantee is `violated`, when a service is in a `not_ready` state. The only admissible combinations are the following ones:

(1) (`not_ready`, `not_determined`)
(2) (`ready`, `not_determined`)
(3) (`running`, `fulfilled`)
(4) (`running`, `violated`)
(5) (`finished`, `fulfilled`)
(6) (`finished`, `violated`)

In theory, there are 63 possible combinations of states in which terms can be. That is, $\sum_{i=1}^{6} \binom{6}{i}$ all terms could be in state (1), or in state (2),... or in state (6); there could be terms in states (1) and (2), (1) and (3), and so on. But again, considering the definition of WS-Agreement in [4], one concludes that not all 63 combinations make sense. Furthermore, it is possible to extract the possible evolutions of these aggregated internal states.

© 2006 ACADEMY PUBLISHER

When an agreement is created its external state is `not_observed`, while all services are `not_ready` and all guarantees are `not_determined`, i.e., state (1). In the next stage some services will be `ready` while others will still be `not_ready`, i.e., there will be terms in state (1) and (2). In this case, the external state is also `not_observed`. Proceeding in this analysis, one can conclude that there are 8 situations in which terms can be. We summarize these in the table in Fig. 2. In the table, we also present the relation between the internal states and the external states, and the set of transitions to go from one set of states to another. The latter transitions are best viewed as an automaton (which is illustrated in [36]).

| | terms' state | agreement's state | transitions |
|---|---|---|---|
| A | (1) | `not_observed` | B |
| B | (1)(2) | `not_observed` | C E |
| C | (1)(2)(3) | `observed` | D E F G |
| D | (1)(2)(3)(5) | `observed` | F G |
| E | (1)(2)(4) | `observed` | F H |
| F | (1)(2)(3)(4)(5) | `observed` | H |
| G | (5) | `finished` | |
| H | (1)(2)(3)(4)(5)(6) | `finished` | |

Fig. 2. Relation between internal and external states.

## V. Extension of WS-Agreement

From the semantics and formal analysis presented in Section IV, inspecting the automaton provided, we note that if the agreement arrives into the states (E) or (F) there is a non recoverable failure, and consequently an agreement termination. Even if one single term is violated, the whole agreement is terminated. Furthermore, when an agreement is running there is no consideration on *how* the guarantee terms are fulfilled. Our goal is to provide an extension of WS-Agreement and of its semantics in order to make agreements more long-lived, and robust to individual term violations. In [37] we provide appropriate XML syntax to implement the proposed extension, while an example of using a subset on a concrete case study (DeltaDator Spa, Trento) of the proposed extension can be found in [36].

We propose to extend WS-Agreement. On the one hand, one can **(i) anticipate violations**; on the other hand, the **(ii) negotiation** of the SLA should be part of its life-cycle. In particular, there is an **(ii.a) initial negotiation** before the execution of the services under SLA, and a **(ii.b) run-time renegotiation** which occurs in case of a recoverable violation of a term or in case the monitoring system is anticipating a possible violation of a term. **(i)** In WS-Agreement guarantees of a running service are either fulfilled or violated. Nothing is said about how a guarantee is fulfilled. Is the guarantee close or far from being violated? Is there a trend bringing the guarantee close to its violation? We propose to introduce a new state for the agreement in which a warning has been issued due to the fact that one or more guarantees are likely to be violated in the near future. By detecting possible violations, one may intervene by modifying the run-time conditions or might renegotiate the guarantees which are close to being violated. **(ii)** The negotiating phase occurs in two moments of the life-cycle of the agreement. In the initial phase the service provider and consumer, must agree on what the conditions of the agreement are. The WS-Agreement specification does not focus on parties involved in the agreement process interactions leading to negotiation of QoS parameters, at most one can use pre-compiled templates. Furthermore, during the execution of the services under agreement, renegotiation may occur when conditions vary or terms are violated or could be violated in the near future. The WS-Agreement specification does not contemplate changing an agreement at run-time, i.e., renegotiation. If a guarantee is not fulfilled because of resource overload or faults in assigning available resources to consumers, the agreement must terminate. For maintaining the service and related supplied guarantees, it is necessary to negotiate the QoS again and create another agreement. This approach wastes resources and computational time, and increases network traffic. The goal of negotiation terms applying is to have the chance to modify the agreement rather than respecting the original agreement. Applying the negotiation terms means that the services included in the agreement will be performed according to the new guarantees.

### A. Life-cycle and semantics for the extended agreement

To obtain the desired extensions, we expand the set of states in which an agreement and a guarantee term can be and thus update the transition system. More precisely, the definition of an agreement does not change with respect to Definition 2, the difference lies in the fact that the set of terms $T$ is now extended with special *negotiation terms*. These terms are defined as in Definition 1, but have a different role, i.e., they specify new conditions that enable modification of guarantees at run-time.

To account for the new type of terms, we need to extend the definition of external and internal state of an agreement. The external states of an extended agreement are enriched by the `negotiated` state, the `checked` state, the `warned` state, the `renegotiated` state, and the `denied` state. We say that an agreement can be in one of nine states. `not_observed`, `observed` and `finished` have the same meaning as in WS-Agreement, Fig. 1. An agreement is in the `negotiated` state while the negotiation process. From the `negotiated` state the agreement can go to the `not_observed` state if the agreement is accepted by all the parties or to come abruptly to an end if it is rejected. An agreement is in the `renegotiated` state while the renegotiation process. From the `renegotiated` state the agreement goes to `observed`. An agreement is in state `checked` when the monitoring system is checking its services and guarantees. From the `checked` state the agreement can

© 2006 ACADEMY PUBLISHER

go to five different states: to `finished` if the agreement finishes its life-cycle; to `denied` if the agreement is violated and no negotiation terms can be applied, the agreement must terminate; to `warned` if the monitoring system issues at least one warning for at least one term; back to `observed` if the agreement is fulfilled; to `renegotiated` if the agreement is fulfilled or violated and negotiation terms can be applied.

*Definition 5 (Extended External state):* The *extended agreement external state* $A_{xes}$ of an agreement $A$ is an element of the set {`negotiated`, `not_observed`, `observed`, `warned`, `checked`, `renegotiated`, `denied` or `finished`}.

The transitions between states are illustrated by the automaton in Fig. 3, which is an extension of the one presented in Fig. 1. The automaton represents the new evolution of an agreement where a guarantee are negotiated and can be modified during the processing of a service or a warning can be raised. When a guarantee is violated we have two situations: the first presents a recoverable violation which implies the chance to apply negotiation terms and so the agreement is in the `renegotiated` state, the second presents a non recoverable violation which implies that there is no suitable negotiation term for the current violated guarantee and so the agreement must terminate. Otherwise, if a warning is raised, this can be ignored or the agreement can go in the `renegotiated` state. Also, when a guarantee is fulfilled, it is possible to change the current agreement configuration, applying a negotiation term that changes the QoS.

The internal state definition for the extended agreement is similar to the internal state definition stated before, but a new state for the services is added and three for the guarantees. A new state is `stopped` and is needed to define a state of a service where its associated guarantee is unrecoverable violated and the service must terminate or the guarantee can be renegotiated. It is an intermediate state. A guarantee is `negotiated` while the negotiation or renegotiation process. A guarantee can also be `warned` if it is close to being violated in a given time instant. Other state for a guarantee is the `non_recoverable_violated` state in which a guarantee is violated and it has no related negotiation term for the current violation.

*Definition 6 (Extended Internal state):* The *extended internal state* $A_{xis}$ of an agreement $A$ is a sequence of terms' states $ts_1, \ldots, ts_p$ of maximum size $n \cdot m$, where $ts_i = (ss_j, gs_k)$ represents the state of $g_k$ guarantee with respect to the $s_j$ service. Service and guarantee states range over the following sets, respectively:
$ss_j \in$ {`not_ready`, `ready`, `running`, `stopped`, `finished`}, and
$gs_k \in$ {`not_determined`, `negotiated`, `fulfilled`, `warned`, `violated`, `non_recoverably_violated`}.

As for Definition 4, one notes that not all the state combinations make sense. The only possible ones are the combinations itemized in Section IV plus the following six:

(7) (`ready`, `negotiated`)
(8) (`stopped`, `negotiated`)
(9) (`stopped`, `fulfilled`)
(10) (`stopped`, `violated`)
(11) (`stopped`, `non_recoverably_violated`)
(12) (`running`, `warned`)

Service is ready and guarantee is negotiated, i.e., state combination (7), while initial negotiation process. The state combinations (8), (9), (10) and (11) determine the states when a service is stopped because a guarantee is violated or is being renegotiated. In state (9) a guarantee is fulfilled and we try to improve it applying a negotiation term. In (10) and (11) a guarantee is currently violated. In (10) the service is stopped and the guarantee is violated but it is possible to apply a negotiation term and to preserve the agreement again. In (11), instead, the guarantee is irrecoverably violated and the agreement must terminate, there are not any suitable negotiation term. State (12) represents the fact that a warning has been raised for a running service guarantee.

### B. Framework

The proposed extension to WS-Agreement must be handled by an appropriate framework that allows for monitoring and provides run-time renegotiation.

On the one hand, there must be rules specifying when and how to raise a warning for any given guarantee. These rules should be easy to compute to avoid overloading of the monitoring system and be fast to provide warnings. In addition they should provide good performance in detecting as many violations as possible generating the minimum number of false positives. A forecasting method which enjoys this characteristics is the linear least squares method [38]. The method of linear least squares requires a straight line to be fitted to a set of data points such that the sum of the squares of the vertical deviations from the points to the line is minimized. By analyzing such a parameter of the line as a slope ratio, it is possible to predict a change over time.

On the other hand, to allow for renegotiation of guarantee terms at run-time the parties involved in the agreement need to be able to decide whether a renegotiation has been agreed upon. Before execution it must be possible to specify negotiation terms. This can be done by using appropriate templates in the spirit of the original work in [5].

## VI. Experimental results

We have conducted experimentation to show the feasibility of the warning strategy. We used synthetic data. We generated a sequence of 1100 elements considered as a service guarantee for a single operation over a continuous time interval (for instance the cost of a service which

© 2006 ACADEMY PUBLISHER

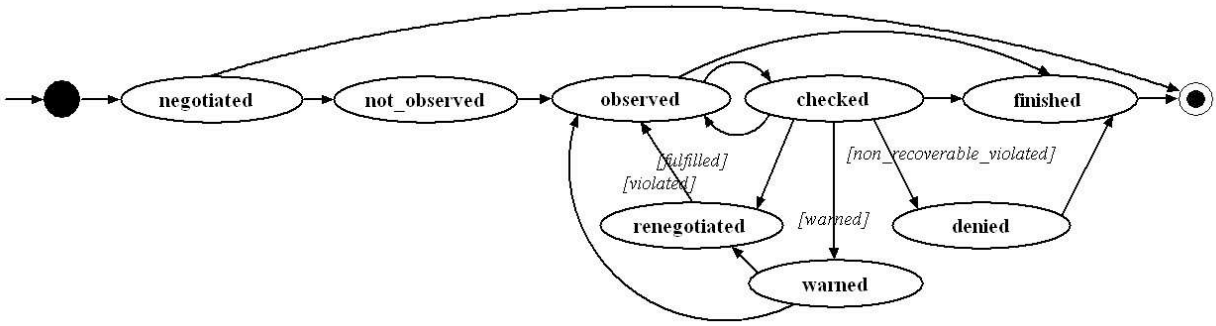


Fig. 3. The life-cycle of the WS-Agreement extension.

should be below the value 10). The data set and the results of the experiments are available in [36]. The points were generated by a function that returns a random number greater or equal to 6.00 and less or equal to 14.00, evenly distributed. We split the data set into two subsets. The first part of the data set was used to decide the size of the time window and of the threshold values to be used for prediction. The rest of the data was used for evaluating the system.

To evaluate the method we consider the following performance measures: *Precision* is the ratio of the number of true warnings (i.e., warnings thrown to notify violation points) to the number of total warnings (i.e., true warnings and false warnings). *Recall* is the ratio of the number of warned violations (i.e., violation points for which a warning is issued) to the number of total violation points. Total violation points include warned violations and missed violations.

| | **Warnings** | | **Violations** | |
|---|---|---|---|---|
| | ***True*** | ***False*** | ***Warned*** | ***Missed*** |
| | 303 | 11 | 156 | 13 |
| **Total** | 314 | | 169 | |
| **Precision** | **96.50%** | | | |
| **Recall** | | | **92.31%** | |

Fig. 4. Experimental results.

The table in Figure 4 summarizes the results of the experimentation. The number of true and false warnings is shown in the first column. The difference in the number of total warnings and violations is due to the fact that more than one warning in the same time window may refer to the same violation. The number of warned and missed violations is reported in the second column of the table. The total sum of warnings and violations is in the "Total" row. The last two rows present the precision and recall of the method.

The results of experimentation on the first 100 points of the data set is shown in Fig. 5. In the figure, two types of warnings, true and false, are marked by diamonds and crosses, respectively. A warning is thrown if the cost and tangent of the cost curve are higher then the threshold

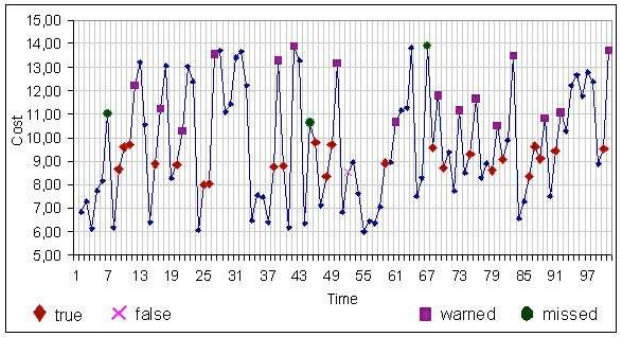


Fig. 5. Experimental results for 100 points.

(8 for cost and 0.1 for the tangent differences). Squares represent warned violation points, while circles indicate missed violation points.

The method shows good performance when the increase in cost is smooth (points 8, 9, and 10), a case that normally takes place during Web Services execution. If the change in values is abrupt then the method fails to generate warnings, e.g., points 43 (cost is 6.36) and 44 (cost is 10.63). It is difficult to find a violation point if the point is in the very beginning of the process, within or just after the first time window (point 7). The latter cases should be considered exceptional, in fact those occur only 13 times in the whole experiment.

In the experimentation using the method, more than 92% of violation points are warned in advance, and 96.5% of thrown warnings are true warnings. Using bigger time windows does not improve performances, see [36] for evidence of this fact.

## VII. Conclusions

WS-Agreement is an industry based protocol for the establishment of service level agreements among loosely coupled service providers and requesters. If on the one hand, WS-Agreement is being adopted widely, on the other hand, it lacks a precise definition of the meaning of its constructs. We presented a formal definition of an WS-Agreement by resorting to finite state automata. Furthermore, by providing a set of formal rules that tie together

© 2006 ACADEMY PUBLISHER

agreement terms and the life-cycle of the agreement, we identified some shortcomings of the protocol. That is, the protocol does not support explicitly the negotiation of the agreement, there is no monitoring of how close a term is to being violated at execution time, and, the breaking of one single term of a running agreement results in termination while a more graceful degradation is desirable. To overcome these shortcomings, we proposed an extension of WS-Agreement, for which we provided appropriate semantics. The extension considers initial negotiation of an agreement, it considers the possibility of issuing warnings before a possible term violation, and eventually renegotiation of a running agreement. We evaluated the approach through simulation experiments on synthetic data. Incidentally, we are not claiming that the proposal based on linear least square is the best approach to provide early warning, in fact, it may turn out that the method depends on the context in which the agreements are established and monitored. Here we are concerned with the extension of the protocol which contemplates the possibility of having early warnings, the way in which these are actually issued will be designed separately for any specific application scenario.

## ACKNOWLEDGMENTS

Marco thanks Asit Dan and Heiko Ludwig for useful discussion on WS-Agreement while visiting IBM TJ Watson. Daniela thanks DIT at the University of Trento for support while completing her bachelor degree.

## REFERENCES

[1] A. Mani and A. Nagarajan, "Understanding Quality of Service for Web Services," IBM developerWorks, January 2002, `http://www-106.ibm.com/developerworks/library/ws-quality.html`.

[2] M. Aiello and P. Giorgini, "Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services," *CEPIS Upgrade - The European journal of the informatics professional*, vol. 5, no. 4, pp. 20–26, 2004.

[3] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid Services for Distributed System Integration," *IEEE Computer*, vol. 35, no. 6, pp. 37–46, 2002.

[4] 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)," Grid Resource Allocation and Agreement Protocol (GRAAP) WG, September 2005, `http://www.ggf.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf`.

[5] H. Ludwig, A. Dan, and R. Kearney, "CREMONA: an Architecture and Library for Creation and Monitoring of WS-Agreements," in *Proceedings of the Second International Conference on Service-Oriented Computing*, New York City, NY, USA, November 2004.

[6] S. Ran, "Model for Web Services Discovery with QoS," *SIGEcom Exchanges*, vol. 4, no. 1, pp. 1–10, 2004.

[7] S. Frølund and J. Koistinen, "Quality-of-Service Specification in Distributed Object Systems," *Distributed Systems Engineering*, vol. 5, no. 4, pp. 179–202, December 1998.

[8] A. Atzeni and A. Lioy, "Why to Adopt a Security Metric? A Brief Survey," in *Proceedings of the First Workshop on Quality of Protection*, Milan, Italy, September 2005.

[9] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Malhotra, A. Nadalin, N. Nagaratnam, H. P. M. Nottingham, C. von Riegen, J. Schlimmer, C. Sharp, and J. Shewchuk, "Web Services Policy Framework (WS-Policy)," March 2006, `http://www-128.ibm.com/developerworks/library/specification/ws-polfram/`.

[10] H. Ludwig, "Web Services QoS: External SLAs and Internal Policies or How do we Deliver what we Promise?" in *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December 2003.

[11] D. Gouscos, M. Kalikakis, and P. Georgiadis, "An Approach to Modeling Web Service QoS and Provision Price," in *Proceedings of the First Web Services Quality Workshop at WISE*, Rome, Italy, December 2003.

[12] M. Tian, A. Gramm, H. Ritter, and J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web Services with the WS-QoS Framework," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, Beijing, China, September 2004.

[13] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality Driven Web Services Composition," in *Proceedings of the 12th International conference on World Wide Web*, Budapest, Hungary, May 2003.

[14] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint Driven Web Service Composition in METEOR-S," in *Proceedings of the 2004 IEEE International Conference on Services Computing*, Shanghai, China, September 2004.

[15] M. Lin, J. Xie, H. Guo, and H. Wang, "Solving QoS-Driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction," in *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service*, Hong Kong, China, March-April 2005.

[16] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, "A Quality of Service Management Framework Based on User Expectations," in *Proceedings of the First International Conference on Service-Oriented Computing*, Trento, Italy, December 2003.

[17] E. Maximilien and M. Singh, "A Framework and Ontology for Dynamic Web Services Selection," *IEEE Internet Computing*, vol. 08, no. 5, pp. 84–93, September/October 2004.

[18] A. S. Bilgin and M. P. Singh, "A DAML-Based Repository for QoS-Aware Semantic Web Service Selection," in *Proceedings of the Second International Conference on Web Services*, San Diego, California, USA, July 2004.

[19] O. Martín-Díaz, A. R. Cortés, A. Durán, D. Benavides, and M. Toro, "Automating the Procurement of Web Services," in *Proceedings of the First International Conference on Service-Oriented Computing*, Trento, Italy, December 2003.

[20] J. Trienekens, J. Bouman, and M. van der Zwan, "Specification of Service Level Agreements: Problems, Principles and Practices," *Software Quality Journal*, vol. 12, no. 1, pp. 43–57, March 2004.

[21] C. Molina-Jimenez, J. Pruyne, and A. van Moorsel, "The Role of Agreements in IT Management Software," *Architecting Dependable Systems III*, pp. 36–58, 2005.

[22] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification. Version 1.0," IBM Corporation, January 2003, `http://www.research.ibm.com/wsla/`.

[23] D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," in *Proceedings of the 9th IEEE Workshop on Future Trends*

© 2006 ACADEMY PUBLISHER

*of Distributed Computing Systems*, San Juan, Puerto Rico, May 2003.
[24] V. Tosic, "WSOL Version 1.2," Department of Systems and Computer Engineering, Carleton University, Tech. Rep. SCE-04-11, July 2004.
[25] C. Molina-Jimenez, S. Shrivastava, E. Solaiman, and J. Warne, "Contract Representation for Run-time Monitoring and Enforcement," in *Proceedings of the IEEE International Conference on E-Commerce Technology*, Newport Beach, California, USA, June 2003.
[26] H. Gimpel, H. Ludwig, A. Dan, and B. Kearney, "PANDA: Specifying Policies for Automated Negotiations of Service Contracts," in *Proceedings of the First International Conference on Service Oriented Computing*, Trento, Italy, December 15-18 2003.
[27] A. Elfatatry and P. Layzell, "A Negotiation Description Language," *Software - Practice and Experience*, vol. 35, no. 4, pp. 323–343, April 2005.
[28] H. Demirkan, M. Goul, and D. S. Soper, "Service Level Agreement Negotiation: A Theory-based Exploratory Study as a Starting Point for Identifying Negotiation Support System Requirements," in *Proceedings of the 38th Hawaii International Conference on System Sciences*, Big Island, HI, USA, January 3-6 2005.
[29] S. Michael, *Engineering Electronic Negotiations*. New York: Kluwer Academic Publishers, 2002.
[30] D. M. Reeves, M. P. Wellman, and B. N. Grosof, "Automated Negotiation from Declarative Contract Descriptions," in *Computational Intelligence*, vol. 18, no. 4, November 2002, pp. 482–500.
[31] C. Molina-Jimenez, S. Shrivastava, J. Crowcroft, and P. Gevros, "On the Monitoring of Contractual Service Level Agreements," in *Proceedings of the First IEEE International Workshop on Electronic Contracting*, San Diego, CA, USA, July 2004.
[32] A. Keller and H. Ludwig, "Defining and Monitoring Service Level Agreements for Dynamic e-Business," in *Proceedings of the 16th USENIX System Administration Conference*, Philadelphia, PA, USA, November 2002.
[33] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu, "Agreement-based Grid Service Management (OGSI-Agreement)," Global Grid Forum, GRAAP-WG Author Contribution, Tech. Rep., 2003.
[34] A. Dan, K. Keahey, H. Ludwig, and J. Rofrano, "Guarantee Terms in WS-Agreement," Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, Tech. Rep., 2004.
[35] A. Andrieux, A. Dan, K. Keahey, H. Ludwig, and J. Rofrano, "Negotiability Constraints in WS-Agreement," Grid Resource Allocation Agreement Protocol (GRAAP) Working Group Meetings, Tech. Rep., 2004.
[36] M. Aiello, G. Frankova, and D. Malfatti, "What's in an Agreement? A Formal Analysis and an Extension of WS-Agreement," DIT, University of Trento, Tech. Rep. DIT-05-039, 2005.
[37] D. Malfatti, "A Framework for the Monitoring of the QoS by extending WS-Agreement," Master's thesis, Corso di Laurea in Informatica, Università degli Studi di Trento, 2005, in Italian.
[38] R. K. Bock, *The Data Analysis: Briefbook*. Springer: Berlin [etc.], 1998.

**Ganna Frankova** received the bachelor's and master's degrees in Computer Science from the Dnepropetrovsk State University, Ukraine, in 2003 and 2004, respectively.

From November 2004 she is a doctorate student of the University of Trento, Italy under the supervising of Dr. Marco Aiello and Prof. Fabio Massacci. Her research interests lie in the field of Service-Oriented Computing with a particular focus on QoS and Security.

Ms. Frankova is a member of the ACM.

**Daniela Malfatti** received the bachelor's degree in Computer Science from the University of Trento in 2005 and is currently pursuing the master's degree in the same university.

Her research interests are in the field of Distributed Systems and Service-Oriented Computing.

**Marco Aiello** received the PhD from the University of Amsterdam in 2002 on Spatial Reasoning. He holds a degree in Engineering and Computer Science from the University of Rome La Sapienza, awarded cum laude in 1997.

He is an assistant professor at the University of Trento, currently on sabbatical leave at the Distributed Systems Group at TUWien. His research interests lie in the fields of Service-Oriented Computing, Document Image Understanding and Spatial Reasoning.

Dr. Aiello is a member of ACM, IEEE, and AI*IA. In 2003 he won the best dissertation award from the Italian Association for Artificial Intelligence (AI*IA).

© 2006 ACADEMY PUBLISHER