A FRAMEWORK FOR PROACTIVE SLA NEGOTIATION

Khaled Mahbub and George Spanoudakis
Department of Computing, City University London, Northampton Square, London EC1V 0HB, U.K.

Keywords: Service Discovery, Service Level Agreements, Proactive SLA Negotiation, Service Monitoring.

Abstract: In this position paper we propose a framework for proactive SLA negotiation that integrates this process with dynamic service discovery and, hence, can provide integrated runtime support for both these key activities which are necessary in order to achieve the runtime operation of service based systems with minimised interruptions. More specifically, our framework discovers candidate constituent services for a composite service, establishes an agreed but not enforced SLA and a period during which this pre-agreement can be activated should this become necessary.

1 INTRODUCTION

A service level agreement (SLA) is an explicit contract between the provider and the consumers of a service that defines the quality and, sometimes, functional properties which should be guaranteed during the provision of the service, as well as the penalties that should be applied in case of defaulting (Wieder, et al., 2008). An SLA is set through a negotiation process between the provider and the consumer of a service. This process is particularly complex in the case of composite services since, in order to ensure that the provision of a composite service S is in line with the SLAs required by its clients, the provider of S should also negotiate and establish subordinate SLAs with the providers of the constituent services of S. Furthermore, when a constituent service of S becomes unavailable at runtime or fails to perform according to its SLA, the provider of S should be able to discover alternative replacement services for it and negotiate SLAs with them at runtime.

As it has been suggested by Zisman, et al. (2008), to minimise the runtime interruption in the provision of composite services, the discovery of back up replacement services for their constituents should be proactive, i.e., it should be performed before a constituent service of S becomes unavailable or fails to perform according to its established SLA. Proactiveness is important since service discovery is a time consuming activity and, therefore, carrying it in a reactive mode, is likely to cause significant interruption in the provision of the composite service and violations of its own SLAs. SLA negotiation should also be proactive as it will be necessary to have adequate SLAs for the potential replacement services that have been identified by proactive discovery attempting SLA negotiation just prior to binding to an alternative service is likely to cause significant delay.

Existing work on service level agreements has focused on SLA specification (Kritikos and Pernici, 2009), negotiation (Dumitrescu and Foster, 2005) and monitoring (Mahbub and Spanoudakis, 2007; Raimondi, et al., 2007). The need for runtime SLA negotiation or re-negotiation has been realised by Di Modica, et al. (2007); Parkin, et al. (2008); Sakellariou and Yarmolenko (2005) and He, et al. (2009), where either the terms of an SLA are revised to accept service from an existing provider (Di Modica, et al., 2007; Sakellariou and Yarmolenko, 2005) or a new SLA is negotiated with a new service provider and an existing SLA is terminated (Parkin, et al., 2008). All these approaches, however, are reactive as they support corrective actions only after an SLA has been violated. Thus, they may fail to guarantee uninterrupted runtime provision of composite services.

To address this shortcoming, in this position paper we introduce an approach for proactive runtime SLA negotiation. Our approach is based on an extension of a tool for proactive runtime service discovery which is described by Zisman, et al. (2008). Our approach weaves SLA negotiation into runtime service discovery and provides a clear process model for carrying these two activities in a

Mahbub K. and Spanoudakis G. (2010).
A FRAMEWORK FOR PROACTIVE SLA NEGOTIATION.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 251-256
DOI: 10.5220/0003040902510256
Copyright © SciTePress

coordinated manner. It also extends a language for expressing runtime service discovery queries that has been developed by Zisman, et al. (2008) to enable the specification of SLA negotiation criteria using it. Thus, our approach provides integrated runtime support for both proactive service discovery and SLA negotiation that enables provision of composite service with minimised interruptions.

Proactive SLA negotiation is weaved into the discovery process and is performed after the execution of service discovery queries to ensure that adequate SLAs can be set for the discovered services. The objective of proactive negotiation is to establish an agreed but not enforced SLA and a period during which the consumer of the service will be able to activate the pre-agreement should this become necessary. The negotiation process is also repeated when a pre-agreed SLA comes close to expiry and, therefore, it has to be renegotiated.

The rest of this paper is structured as follows. In Section 2, we discuss the architecture of the framework for proactive runtime service discovery and SLA negotiation. In Section 3, we describe the negotiation process. In Section 4, we briefly present the negotaiton mechanism used in the farmewrok. In Section 5, we review related work and finally in Section 6, we provide some concluding remarks and outline directions for future work.

2 PROACTIVE SERVICE DISCOVERY AND SLA NEGOTIATION FRAMEWORK

The architecture of our integrated service discovery and SLA negotiation framework is shown in Figure 1. The framework consists of a runtime service discovery tool, a service listener, a proxy negotiation broker and a monitor, and interacts with external service registries and event captors.

The runtime service discovery tool is used to identify potential alternative services for the services that a composite service uses currently. The discovery process is driven by service discovery queries. These queries are associated with each of the constituent services S_c of the composite service S and specify the conditions that should be satisfied by any service that could replace S_c's in the composition. These conditions can refer to the structural (interface), behavioural, contextual, and quality characteristics that services should have in order to be acceptable replacements for S_c. Service discovery queries can be executed in two modes: (a) in a reactive mode where the query is executed when the constituent service S_c that it is associated with becomes unavailable or fails to satisfy an agreed SLA and, therefore, a replacement service should be identified, or (b) in a proactive mode where the query is executed in parallel with the operation of the composite service S in order to discover and maintain a set of candidate replacement services for it. In the proactive execution mode, the query is executed initially to build a set of replacement services for S (RS) and then anytime when an event indicating that the description of some service in RS has been changed or a new service that could be a candidate for inclusion in RS has emerged.

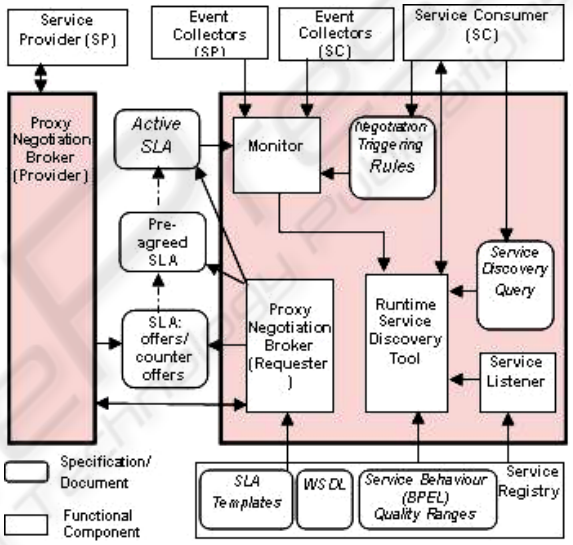

Figure 1: Architecture for proactive (and reactive) SLA negotiation.

The *proxy negotiation broker* is the component that manages the negotiation process on behalf of a service consumer (i.e., the composite service) or a service provider. Our architecture assumes that a separate instance of this component is associated with each of the two sides (the service provider and consumer). The structure of proxy negotiation broker is described in more detail in Section 2.1. The negotiation process can be either reactive or proactive. In proactive negotiation, the negotiation process is carried out according to a two-phase protocol that may result in a provisionally agreed SLA but not activated SLA (see "Pre-agreed SLA" in Figure 1) or negotiation failure. In reactive negotiation, the negotiation process is executed according to a single phase protocol that can result in an agreed and activated SLA (see "Active SLA" in Figure 1) or negotiation failure. In the framework, a pre-agreed SLA describes a service level

agreement that has been reached but not activated yet. A pre-agreed SLA has an expiry period within which it becomes active, if the consumer of the service decides to activate it.

The service registry contains descriptions of services. These should include at least a specification of the interface of the service (WSDL) and SLA templates indicating the terms (e.g. service quality levels, costs etc) under which the provider of service is typically willing to provide it. Additional types of service descriptions that are supported by the framework are models of service behavior (expressed in BPEL) and further quality characteristics that complement the SLA templates by specifying the entire range of values for a given characteristic.

The service listener contacts service registries regularly to identify changes in existing service descriptions or new services that might have become available.

The monitor in the architecture of Figure 1 is responsible for monitoring the provision of a service by a given provider and the use of it by a set of service consumers. The monitor is used to detect if the SLA guarantee terms which should apply to the provision of the service are satisfied, and whether the conditions of the negotiation triggering rules of the relevant party are satisfied in order to generate signals for triggering negotiation.

If monitor detects (or forecasts) that the conditions of negotiation triggering rules in the negotiation policy of a service provider or consumer are (or will be) violated, it informs the relevant negotiation broker to initiate a negotiation or renegotiation. The checks performed by the monitors take into account events that are intercepted during the use of services (e.g. service invocations and responses). These events are notified to the framework by different types of event captors that may be associated with different services (e.g. SOAP message captors). These events are notified to the monitor for verifying the adherence of services to different SLA guarantee terms and checking whether some SLA negotiation activity should be initiated.

The circumstances under which the negotiation of new SLAs should start are determined by *negotiation triggering rules* (e.g., when a provisionally agreed SLA is about to expire). Service providers and consumers may specify separate sets of such rules, which come to force and get monitored after the SLA is established.

2.1 Proxy Negotiation Broker

Figure 2, shows the structure of the proxy negotiation broker. The broker deploys suitable *Negotiation Engines* that are responsible for negotiating and agreeing the guarantee terms of an SLA. The broker wraps different types of negotiation engines (e.g. rule driven negotiation engine or linear programming based negotiation engine) and provides a common interface to access its functionalities. The *Specification Translator* is responsible for generating specifications that are necessary for the negotiation engine to conduct the negotiation process.

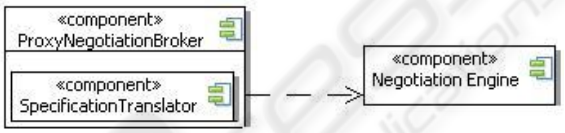

Figure 2: Proxy negotiation broker and its components.

Figure 3 shows the interface of the proxy negotiation broker. This interface offers the following methods,

- *SetupNegotiation*: This method allows the service discovery process to initialize the negotiation broker for the negotiation process. This method accepts four parameters. The first parameter is the string representation of an XML document that contains the negotiation rules. The XML document is written according to the negotiation rule specification schema of our framework (Mahbub and Spanoudakis, 2010). The specification translator transforms these negotiation rules into a form that is understood by the deployed negotiation engine. The second parameter specifies the type of negotiation engine that should be used in the negotiation process (e.g. rule based or linear programming based). The third parameter specifies the ID of the service consumer that the negotiation broker is negotiating for and the fourth parameter specifies the ID of the negotiation broker that this negotiation broker is negotiating with. This method returns a unique ID for the negotiation process.
- *StartNegotiation*: This method is used by the service discovery process to start the negotiation process. This method accepts one parameter that specifies the unique ID of the negotiation process.
- *SetOffers*: The *Proxy Negotiation Broker* uses this method to set the offers produced by the negotiation engine to its counterpart, i.e., the

negotiation broker on the other side that this *negotiation broker* is negotiating with. This method accepts one parameter, which is the string representation of an XML document that contains an SLA template. The XML document is written according to the SLA specification schema of our framework (Mahbub and Spanoudakis, 2010).

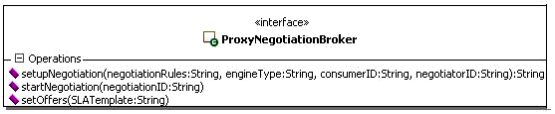

Figure 3: Interface of proxy negotiation broker.

3 SLA NEGOTIATION PROCESS

The UML activity diagram shown in Figure 4 presents the service discovery process of the framework with the activity of SLA negotiation embedded within it. The process starts with the submission of a service discovery query by the composite service (i.e., the consumer of constituent services). As discussed in Section 2, this query can specify different service discovery criteria, namely: (a) structural criteria describing the interface of required services, (b) behavioural criteria describing the functionality of required services, and (c) constraints describing quality characteristics of required service. The initial execution of the service discovery query (see the action state *Execute Query* in Figure 4) results in a list of potential candidate services (RS). The candidate services are identified by evaluating the structural, behavioural and quality characteristics specified in a query against the structural, behavioural and quality of service specifications in service registries. The execution of the discovery query also computes distances between a query and candidate services based on the query criteria and ranks the candidate services based on their distances to the query. The list of potential candidate services is updated by executing the service discovery query when the framework is informed via the service listener that a new service has become available in a registry or the description of an existing service has been modified (see the signal accept state *New/Amended Service Description* in Figure 4). This ensures that new or updated services are considered by the process.

After an initial set of candidate services has been built or updated (see the action state *Create/Update Candidate Service Set*), the framework selects a service that does not have a negotiated SLA from RS for negotiation (see the transition guarded by the condition *Exists Service in RS without Negotiated SLA*).

In the negotiation phase (i.e., the action state *Negotiate SLA*), the desired level of service is negotiated with the selected candidate service. In this phase, the QoS characteristics of each candidate service are negotiated in order to achieve the best possible SLA for the services. Negotiation during this phase may fail and, if this happens, for a selected candidate service then the service is removed from RS and a new negotiation will start with another candidate service in RS which does not have a negotiated SLA. If the negotiation with a selected service succeeds, a provisional SLA is established and the selected service in RS is updated to flag the existence of the pre-agreed SLA.

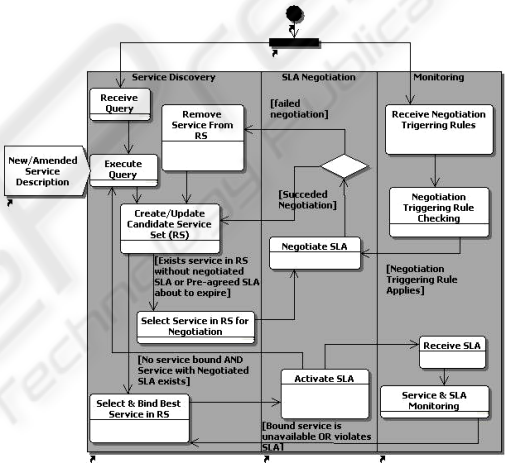

Figure 4: SLA negotiation process.

It should be noted that, the negotiated SLAs for the services in RS do not come into force immediately. For each pre-agreed SLA, the negotiation process establishes a time period over which the pre-agreed SLA can be automatically brought into force without further negotiation. This will happen if the relevant service is selected for binding to the composite service. If the validity period of a pre-agreed SLA comes close to expiry without the candidate service being bound to the composite service, the framework will proactively re-negotiate the SLA (see the transition guarded by the condition *Pre-agreed SLA about to expire*, from the action state *Create/Update Candidate Service Set* to the action state *Select Service RS for Negotiation*).

Following the selection of a service in RS for binding at runtime, its SLA is automatically enforced (see the action state *Activate SLA* in Figure 4). When an SLA comes into force, its guarantee

terms become subject of monitoring (see the action states *Receive SLA* and *Service & SLA Monitoring* in Figure 4). If the monitoring process detects violation of the SLA or the deployed service becomes unavailable then the service is replaced by the best available service in RS (see the transition from the action state *Service & SLA Monitoring* to the action state *Select & Bind Best Service in RS*). The detection of violation of the conditions in negotiation triggering rules (e.g. active SLA about to expire) triggers the negotiation phase to establish a new SLA.

4 NEGOTIATION MECHANISM

Although, as discussed in Section 2, our framework supports the deployment of different types of negotiation engines, we use a linear programming based negotiation engine as the default. This is because linear programming enables relatively quick optimal solution in decision making (Megiddo 1987). Despite this advantage, the use of linear programming techniques in the context of SLA negotiation has received limited attention, and the few approaches that apply linear programming for SLA management have certain limitations. For example, Hung (2003) assumed that participating parties expose their preferences over the negotiation issues in order to formulate the linear program. In reality, however, participants might not be willing to disclose information about their own preferences. Cradellini, et al. (2007) used linear programming to set and tune the provision of an agreed SLA rather than in SLA negotiation.

In our framework, each participant expresses its preferences over negotiation issues by a set of negotiation rules. The proxy negotiation broker transforms the negotiation rules into a linear program and solves it to generate SLA offers during the negotiation process. A negotiation rule in our framework has the generic structure:

IF (condition) THEN (action) ELSE (action)

Conditions in these rules are expressed as atomic conditions over quality attributes of services or logical combinations of atomic conditions. Rule actions can be of three types: (i) *accept actions* which enable the acceptance of the value of one or more attributes in a given SLA offer, (ii) *reject actions* which enable the rejection of the value of one or more QoS attributes in a given SLA offer, and (iii) *set actions* which allow to define a new value or range of values for one or more QoS attribute.

Negotiation rules are transformed into linear inequalities applying the techniques discussed by Nielsen, et al. (2000); Costa and Monteiro (2003). For example, a negotiation rule of the form,

IF (p > 0) THEN (q >=0)

can be transformed into the following set of inequalities,

```
0 < P + C1*B
0 <= Q + C2*B
P <= 0 + C3*(1-B)
```

where, B is a binary variable and C1, C2, C3 are constants whose values should be large enough.

5 RELATED WORK

There is a substantial amount of work related to SLA negotiation. The parts of it that are more relevant to our approach relate to runtime negotiation and management of compensations for SLA violations.

The provision of compensation in case of violation of SLA is argued by Rana, et al. (2007) and He, et al. (2009). These approaches claim that the penalty clauses in the SLA should not only specify the monetary penalties or impact on potential future agreements between the parties but also several other issues such the law that will be applied in cases where a conflict between the provider and the client arise, and the impact of the penalty clauses on the choice of service level objectives.

Runtime renegotiation is suggested by Wieder, et al. (2008); Sakellariou and Yarmolenko (2005); Di Modica, et al. (2007) and Parkin, et al. (2008), to manage SLA violations. Service level objectives are revised and renegotiated at runtime and the deployed service is adjusted to the newly agreed service level objectives by Di Modica, et al. (2007). A similar approach which allows changes in service level objectives whilst keeping an existing SLA is described by Sakellariou and Yarmolenko (2005). Parkin, et al. (2008) described a renegotiation protocol that allows the service consumer or service provider to initiate renegotiation while the existing SLA is still in forced. In this protocol any party may initiate the renegotiation and after a successful renegotiation the existing SLA is superseded by a new contract.

The approaches discussed above are reactive in nature, i.e., renegotiation starts only after an existing SLA is violated. The outcome of renegotiation is either a revised set of service level objectives

allowing the acceptance of a service from an existing provider or a new SLA for a new service provider terminating the existing SLA. Hence, all these approaches either affect the quality of the delivered service or fail to guarantee uninterrupted service. Unlike them, our approach weaves proactive SLA negotiation into dynamic service discovery to enable the runtime operation of service based applications with minimised interruptions.

6 CONCLUSIONS AND FUTURE WORK

This position paper proposes a framework that integrates service discovery, monitoring and proactive SLA negotiation. The service discovery process is used by service consumers (i.e., service based applications) in order to identify potential alternative services for the services that they currently use. The framework supports the proactive negotiation of SLAs with each alternative service prior to its deployment. The negotiation process is carried out according to a two-phase protocol that may result in a provisionally agreed SLA but not activated SLA or negotiation failure. A provisional SLA is a service level agreement that has been agreed by a service provider and a service consumer but has not been activated yet. Such an SLA has an expiry date by which it will either be activated or cease to exist.

The presented framework has also opened broad scope of future investigations. For example the framework can be extended to support proactive negotiation for hierarchical SLA i.e. a complex SLA can be decomposed into several SLAs and negotiated separately to come to a final agreement. Also the framework can be extended to support dynamic adaptation of the negotiation rules, i.e. the participants will be able to dynamically change the negotiation rules during the negotiation process.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under Grant Agreement 215483 (S-Cube).

REFERENCES

Cardellini V., Casalicchio E., Grassi V., Lo Presti F., 2007. Efficient provisioning of service level agreements for service oriented applications. *In Proceedings of IW-SOSWE at ESEC/FSE.*

Costa J. C., Moneiro J. C., 2003. Input Generation for Path Coverage in Software Testing. *IEEE Workshop on Compilers and Tools for Constrained Embedded Systems*

Di Modica G., Tomarhio O. and Lorenzo V., 2007. A framework for the management of dynamic SLAs in composite service scenarios. *ICSOC Int. Workshops,*

Dumitrescu C. L., Foster I., 2005. GRUBER: A Grid Resource Usage SLA Broker. *Int. Euro-Par conference.*

He Q., Yan, J., Kowalczyk R., Hai J., Yang Y., 2009. Lifetime Service Level Agreement Management with Autonomous Agents for Services Provision. *Information Sciences,* Elsevier.

Hung P. C. K., 2003. A Primitive Study of Logrolling in e-Negotiation. *36thHawaii Int. Conf. on System Sciences*

Kritikos K., Pernici B., (editors), 2009. Initial Concepts for Specifying End-to-End Quality Characteristics and Negotiating SLAs. *S-Cube project deliverable: CD-JRA-1.3.3. http://www.s-cube-network.eu/achievements-results/s-cube-deliverables.*

Mahbub K., Spanoudakis G., 2007. Monitoring WS Agreements: An Event Calculus Based Approach, *Test and Analysis of Service Oriented Systems*, (eds) L. Baresi, E. diNitto, Springer.

Megiddo N., 1987. On the complexity of linear programming. *In Advances in economic theory Fifth world congress,* Cambridge University Press

Nielsen S. R., Pisinger D., Marquardsen P., 2000. Automatic Transformation of Constraint Satisfaction Problems to Integer Linear Form - an Experimental Study. *In Proc. of Techniques foR Implementing Constraint programming Systems,*

Parkin M., Hasselmeyer P., Koller B., Wieder P., 2008. An SLA Re-negotiation Protocol. In *proceedings of the 2nd Workshop on Non Functional Properties and SLAs in Service Oriented Computing*

Raimondi, F., Skene, J., Chen L., Emmerich, W., 2007. Efficient monitoring of web service SLAs. *Technical report. Research Notes (RN/07/01). UCL, London,.*

Rana O., Warnier M., Quillinan T. B., Brazier F. and Cojocarasu D., 2007. Managing Violations in Service Level Agreements. In *Proceedings of the Usage of Service Level Agreements in Grids Workshop.*

Sakellariou R., Yarmolenko V., 2005. On the Flexibility of WS-Agreement for Job Submission. *Proc. of 3rd Int. Workshop on Middleware for Grid Computing,*

Mahbub K, Spanoudakis G., 2010. SLA Specifications. http://www.soi.city.ac.uk/~am697/sla/SLA-Spec.zip

Wieder P., Seidel J., Wäldrich O., Ziegler W., Yahyapour R., 2008. Using SLA for Resource Management and Scheduling - A Survey. *Grid Middleware and Services Challenges and Solutions,* Springer.

Zisman A., Spanoudakis G., Dooley J., 2008. A Framework for Dynamic Service Discovery. *23rd Int. Conf. on Automated Software Engineering.*