SERVICE MEDIATION AND NEGOTIATION BOOTSTRAPPING AS FIRST ACHIEVEMENTS TOWARDS SELF-ADAPTABLE CLOUD SERVICES

Ivona Brandic, Dejan Music, Schahram Dustdar
Institute of Information Systems, Vienna University of Technology
Argentinierstraße 8, 1040 Vienna, Austria
{ivona,dejan,dustdar}@infosys.tuwien.ac.at

Abstract Nowadays, novel computing paradigms as for example Cloud Computing are gaining more and more on importance. In case of Cloud Computing users pay for the usage of the computing power provided as a service. Beforehand they can negotiate specific functional and non-functional requirements relevant for the application execution. However, providing computing power as a service bears different research challenges. On one hand dynamic, versatile, and adaptable services are required, which can cope with system failures and environmental changes. On the other hand, human interaction with the system should be minimized. In this chapter we present the first results in establishing adaptable, versatile, and dynamic services considering negotiation bootstrapping and service mediation achieved in context of the Foundations of Self-Governing ICT Infrastructures (FoSII) project. We discuss novel meta-negotiation and SLA mapping solutions for Cloud services bridging the gap between current QoS models and Cloud middleware and representing important prerequisites for the establishment of autonomic Cloud services.

Keywords: Autonomic SLA Management, Meta-Negotiations, SLA Mapping, Negotiation Frameworks, Cloud Computing

1. Introduction

Service-oriented Architectures (SOA) represent a promising approach for implementing ICT systems [4]. Thereby, software is packaged to services and can be accessed independently of the used programming languages, protocols, and platforms. Despite remarkable adoption of SOA as the key concept for the implementation of ICT systems, the full potential of SOA (e.g., dynamism, adaptivity) is still not exploited [19]. SOA approach and Web service technologies represent large scale abstractions and a candidate concept for the implementation novel computing paradigms where sophisticated scientific applications can be accessed as services over Internet [1, 5] or where massively scalable computing is made available to end users as a service as in case of *Cloud Computing* [10]. In all those approaches the access to computing power is provided as a service.

The key benefits of providing computing power as a service are (a) avoidance of expensive computer systems configured to cope with peak performance, (b) pay-per-use solutions for computing cycles requested on-demand, and (c) avoidance of idle computing resources. The development of novel concepts for dynamic, versatile, and adaptive services represents an open and challenging research issue [16]. Major goal of this chapter is to facilitate service negotiation in heterogeneous Clouds. In order to enable service users to find services which best fit to their needs (considering costs, execution time and other functional and non-functional properties), service users should negotiate and communicate with numerous publicly available services.

Non-functional requirements of a service execution are termed as *Quality of Service (QoS)*, and are expressed and negotiated by means of *Service Level Agreements (SLAs)*. *SLA templates* represent empty SLA documents with all required elements like parties, SLA parameters, metrics and objectives, but without QoS values [12]. However, most existing Cloud frameworks assume that the communication partner knows about the *negotiation protocols* before entering the negotiation and that they have matching *SLA templates*. In commercially used Clouds this is an unrealistic assumption since services are discovered dynamically and on demand. Thus, so-called *meta-negotiations* are required to allow two parties to reach an agreement on what specific negotiation protocols, security standards, and documents to use before starting the actual negotiation. The necessity for SLA mappings can be motivated by differences in terminology for a common attribute such as *price*, which may be defined as *usage price* on one side and *service price* on the other, leading to inconsistencies during the negotiation process.

Thus, we approach the gap between existing QoS methods and Cloud services by proposing an architecture for Cloud service management with components for *meta-negotiations* and *SLA mappings* [9, 8, 7]. Meta-negotiations are defined by means of a *meta-negotiation document* where participating parties may express: the pre-requisites to be satisfied for a negotiation, for example, requirement for a specific authentication method; the supported negotiation protocols and document languages for the specification of SLAs; and conditions for the establishment of an agreement, for example, a required third-party arbitrator. SLA mappings are defined by XSLT[1] documents where inconsistent parts of one document are mapped to another document e.g., from consumer's template to provider's template. Moreover, based on SLA mappings and deployed taxonomies, we eliminate semantic inconsistencies between consumer's and providers SLA template.

2. Related Work

Since there is very little exiting work on self-adaptable Cloud services, we look into existing systems in related areas - in particular into existing Grid systems. Currently, a large body of work exists in the area of Grid service negotiation and SLA-based QoS [20, 11]. Work presented in [22] discusses incorporation of SLA-based resource brokering into existing Grid systems. Glatard et al. discuss a probabilistic model of workflow execution time evaluated in context of EGEE grid infrastructure [13]. Work described in [23] presents an approach for dynamic workflow management and optimization using near-realtime performance with strategies for choosing an optimal service, based on user-specified criteria, from several semantically equivalent Web services. Oldham et al. describe a framework for semantic matching of SLAs based on WSDL-S and OWL [21].

Ardagana et al. [3] present an autonomic Grid architecture with mechanisms to dynamically re-configure service center infrastructures, which is basically exploited to fulfill varying QoS requirements. Work presented in [1] extends the service abstraction in the Open Grid Services Architecture (OGSA) for QoS properties focusing on the application layer. Thereby, a given service may indicate the QoS properties it can offer or it may search for other services based on specified QoS properties. Work presented in [11] proposes a generalized resource management model where resource interactions are mapped onto a well defined set of platform-independent SLAs. The model is based on Service Negotiation and Acquisition Protocol (SNAP) providing the lifetime management SLAs.

Dan et al. [12] present a framework for providing customers of Web services differentiated levels of service through the use of automated management and SLAs. Work described in [15] discusses how semantic technologies may be used by mobile devices which need to locate and select appropriate Grid services

[1]XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt.html

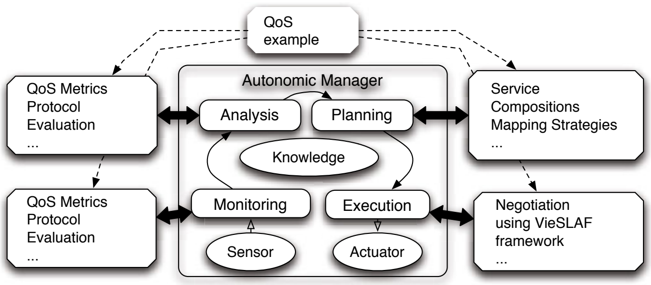
QoS example
QoS Metrics Protocol Evaluation ...
Autonomic Manager
Analysis
Planning
Knowledge
Monitoring
Execution
Sensor
Actuator
Service Compositions Mapping Strategies ...
QoS Metrics Protocol Evaluation ...
Negotiation using VieSLAF framework ...

Figure 1. General Architecture of an Autonomic System Explained on a QoS Example

in an automatic and flexible way. Jurca et al. propose a new form of SLAs where the price is determined by the QoS which is actually delivered by service provider. For the monitoring of QoS a novel approach is introduced based on reputation mechanism [17].

3. Adaptable, Versatile, and Dynamic services

In this section we discuss how adaptable, versatile, and dynamic services can be realized.

3.1 Overview

To facilitate dynamic, versatile, and adaptive IT infrastructures, SOA systems should react to environmental changes, software failures, and other events which may influence the systems' behavior. Therefore, adaptive systems exploiting self-* properties (self-healing, self-controlling, self-managing, etc.) are needed, where human intervention with the system is minimized. We propose models and concepts for adaptive services building on the approach defined by means of autonomic computing [18, 3].

We identified the following objectives:

Negotiation bootstrapping and service mediation. The first objective is to facilitate communication between publicly available services. Usually, before service usage, service consumer and service provider have to establish an electronic contract defining terms of use [6, 11]. Thus, they have to negotiate the exact terms of contract (e.g., exact execution time of the service). However, each service provides a unique negotiation protocol often expressed using different languages, representing an obstacle within the SOA architecture. We propose novel concepts for automatic bootstrapping between different protocols and contract formats increasing the number of services a consumer may negotiate with. Consequently, the full potential of public services could be exploited.

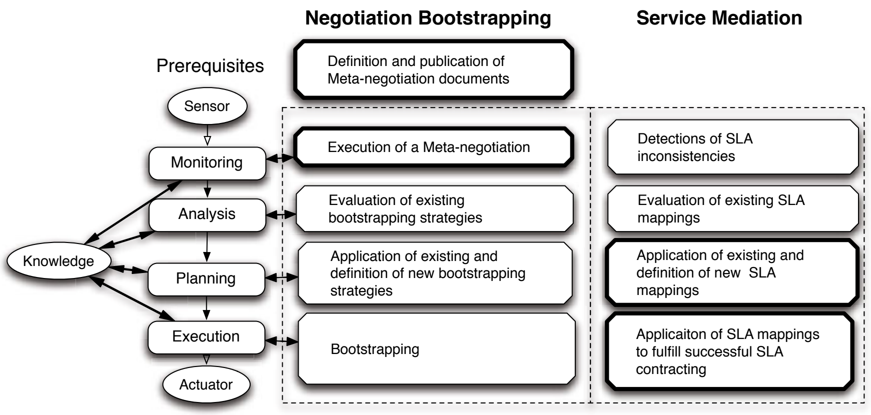
Negotiation Bootstrapping
Service Mediation
Prerequisites
Definition and publication of Meta-negotiation documents
Sensor
Monitoring
Analysis
Planning
Execution
Actuator
Knowledge
Execution of a Meta-negotiation
Evaluation of existing bootstrapping strategies
Application of existing and definition of new bootstrapping strategies
Bootstrapping
Detections of SLA inconsistencies
Evaluation of existing SLA mappings
Application of existing and definition of new SLA mappings
Applicaiton of SLA mappings to fulfill successful SLA contracting

Figure 2. Negotiation Bootstrapping and Service Mediation as Part of the Autonomic Process

Service Enforcement. Services may fail, established contracts between services may be violated. The second objective is to develop methods for service enforcement, where failures and malfunctions are repaired on demand and where services are adapted to changing environmental and system conditions. We propose development of knowledge bases where the directives, policies, and rules for failure adjustment and repair may be specified and stored. Furthermore, adequate methods for the condition specification and condition evaluation are emerging research issues.

Service adaptivity. Service failures or violations of electronic agreements must be detected in an efficient manner. Moreover, the reaction to failures should be done in an adequate way. Thus, the third objective is the development of novel methods for modeling of intelligent logging capabilities at the level of a single service as well as composite services. Sophisticated concepts for the measurement of service execution parameters and Quality of Service (QoS) are needed as well as generic monitoring capabilities which can be customized on-demand for different services.

Service Governance. Policies and rules for service enforcement should not be defined in a static way. Moreover, the rules should evolve over time. The fourth objective is the development of the governing guidelines for rule definition and rule-evolution. This includes the development of adequate languages for rule specification and rule evolution as well as novel reasoning techniques.

In order to achieve aforementioned goals we utilize the principles of *autonomic computing*. Autonomic computing research methodology can be exemplified using Quality of Service (QoS) as shown in Figure 1. The management is done through the following steps: (i) *Monitoring*: QoS managed element is monitored using adequate software sensors; (ii) *Analysis*: The monitored and measured metrics (e.g., execution time, reliability, availability, etc.) are analyzed using knowledge base (condition definition, condition evaluation, etc.); (iii) *Planning*: Based on the evaluated rules and the results of the analysis, the planning component delivers necessary changes on the current setup e.g., renegotiation of services which do not satisfy the established QoS guarantees; (iv) *Execution*: Finally, the planned changes are executed using software actuators and other tools (e.g., *VieSLAF* framework [9]), which query for new services.

3.2 Negotiation Bootstrapping and Service Mediation

Autonomic computing can be applied for other managed elements e.g., service negotiation. In the following we explain the first steps in achieving aforementioned architecture: *meta-negotiations* and *SLA mappings*.

Figure 2 depicts how the principles of autonomic computing can be applied to negotiation bootstrapping and service mediation. As a prerequisite of the negotiation bootstrapping users have to specify a meta-negotiation document describing the requirements of a negotiation, as for example required negotiation protocols, required security infrastructure, provided document specification languages, etc. During the *monitoring phase* all candidate services are selected where negotiation bootstrapping is required. During the *analysis phase* existing knowledge base is queried and potential bootstrapping strategies are found. In case of missing bootstrapping strategies users can define in a semi-automatic way new strategies (*planning phase*). Finally, during the *execution phase* the negotiation is started by utilizing appropriate bootstrapping strategies.

The same procedure can be applied to service mediation. During the service negotiation, inconsistencies in SLA templates may be discovered (*monitoring phase*). During the *analysis phase* existing SLA mappings are analyzed. During the *planning phase* new SLA mappings can be defined, if existing mappings cannot be applied. Finally, during the *execution phase* the newly defined SLA mappings can be applied.

As indicated with bold borders in Figure 2, in this chapter we present solutions for the definition and accomplishment of meta-negotiations (Section 4) and for the specification and applications of SLA mappings (Section 5). In the following section we explain the principles of meta-negotiations.

4. Meta-Negotiations

In this section, we present an example scenario for the meta-negotiation architecture, and describe the document structure for publishing negotiation details into the meta-negotiation registry.

4.1 Meta-Negotiation Scenario

The meta-negotiation infrastructure can be employed in the following manner: (i) *Publishing*: A service provider publishes descriptions and conditions of supported negotiation protocols into the registry; (ii) *Lookup*: Service consumers perform lookup on the registry database by submitting their own documents describing the negotiations that they are looking for. (iii) *Matching*: The registry discovers service providers who support the negotiation processes that a consumer is interested in and returns the documents published by the service providers; (iv) *Negotiation*: Finally, after an appropriate service provider and a negotiation protocol is selected by a consumer using his/her private selection strategy, negotiations between them may start according to the conditions specified in the provider's document.

In the following we explain the sample meta-negotiation document.

4.2 Meta-Negotiation Document (MND)

The participants publishing into the registry follow a common document structure that makes it easy to discover matching documents. This document structure is presented in Figure 3 and consists of the following main sections.

Each document is enclosed within the `<meta-negotiation>` ... `</meta-negotiation>` tags. Each meta-negotiation (MN) comprises three distinguishing parts, namely *pre-requisites*, *negotiation* and *agreement* as described in the following paragraphs.

Pre-requisites. The conditions to be satisfied before a negotiation starts are defined within the `<pre-requisite>` element (see Figure 3, lines 3–10). Pre-requisites define the *role* a participating party takes in a negotiation, the *security credentials* and the *negotiation terms*. The `<security>` element specifies the authentication and authorization mechanisms that the party wants to apply before starting the negotiation process. The negotiation terms specify QoS attributes that a party is willing to negotiate and are specified in the `<negotiation-term>` element. For example, in Figure 3, the negotiation terms of the consumer are *beginTime* and *endTime*, and *price* (line 6).

Negotiation. Details about the negotiation process are defined within the `<negotiation>` element. Each document language is specified within the `<document>` element. In Figure 3, *WSLA* is specified as the supported document language. Additional attributes specify the URI to the API or WSDL for the documents and their versions supported by the consumer. In Figure 3, *AlternateOffers* is specified as the supported negotiation protocol. In addition to the *name*, *version*, and *schema* attributes, the URI to the WSDL or API of the negotiation protocols is specified by the *location* attribute (line 12).

Agreement. Once the negotiation has concluded and if both parties agree to the terms, then they have to sign an agreement. This agreement may be verified

```
<meta-negotiation ...>
 <pre-requisite>
  <role name="consumer"/>
  <security> <authentication value="GSI" location="uri"/> </security>
  <negotiation-terms>
   <negotiation-term name="beginTime"/> <negotiation-term name="endTime"/>
  </negotiation-terms>
 </pre-requisite>
 <negotiation>
  <document name="WSLA" value="uri" version="1.0"/>
  <protocol name="alternateOffers" schema="uri" version="1.0" location="uri"/>
 </negotiation>
 <agreement> <confirmation name="arbitrationService" value="uri"/> </agreement>
</meta-negotiation>
```

Figure 3. Example Meta-negotiation Document

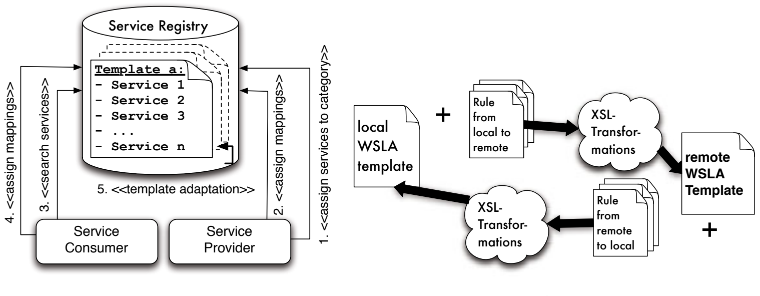

Figure 4. Management of SLA-Mappings (a) Scenario for XSL Transformations (b)

by a third party organization or may be logged with another institution who will also arbitrate in case of a dispute. These modalities are specified within the `<agreement>` clause of the meta-negotiation document as shown in line 14. The meta-negotiation architecture described here was experimentally evaluated and the results were presented in a previous publication [8].

5. SLA mappings

In the presented approach each SLA template has to be published into a registry where negotiation partners i.e., provider and consumer, can find each other.

5.1 Management of SLA mappings

Figure 4(a) depicts the architecture for the management of SLA mappings and participating parties. The registry comprises different *SLA templates* whereby each of them represent a specific application domain, e.g., SLA templates for medical, telco or life science domain. Thus, each service provider may assign his/her service to a particular template (see step 1 in Figure 4(a)) and afterwards assign SLA mappings if necessary (see step 2). Each template *a* may have *n* services assigned.

Service consumer may search for the services using meta-data and search terms (step 3). After finding appropriate services each service consumer may define mappings to the appropriate template the selected service is assigned to (step 4). Thereafter, the negotiation between service consumer and service provider may start as described in the next section. As already mentioned templates are not defined in a static way. Based on the assigned SLA mappings and the predefined rules for the adaptation, SLA templates are updated frequently trying to reflect the actual SLAs used by service provides and consumers (step 5).

Currently, SLA mappings are defined on an XML level, where users define XSL transformations. However, a UML based GUI for the management of SLA mappings is subject of ongoing work [7].

5.2 Scenario for SLA mappings

Figure 4 depicts a scenario for defining XSL transformations. For the definition of SLA agreements we use Web Service Level Agreement (WSLA) [24]. WSLA templates are publicly available and published in a searchable registry. Each participant may download previously published WSLA templates and compare them with the local template. This can be done in an automatic way by using appropriate tools. We are currently developing a GUI that can help consumers to find suitable service categories. If there are any inconsistencies discovered, service consumer may write rules (XSL transformation) from his/her local template to the remote template. The rules can also be written by using appropriate visualization tools. Thereafter, the rules are stored in the database and can be applied during the runtime to the remote template. During the negotiation process, the transformations are performed from the remote WSLA template to the local template and vice versa.

Figure 4 depicts a service consumer generating a WSLA. The locally generated WSLA plus the rules defining transformation from local WSLA to remote WSLA, deliver a WSLA which is compliant to the remote WSLA. In the second case, the remote template has to be translated into the local one. In that case, the remote template plus the rules defining transformations from the remote to local WSLA deliver a WSLA which is compliant to the local WSLA. Thus, in this manner, the negotiation may be done using non-matching templates.

Even the service provider can define rules for XSL transformations from the publicly published WSLA templates to the local WSLA templates. Thus, both parties, provider and consumer, may match on a publicly available WSLA template.

```
...
 <xsl:template ...>
  <xsl:element name="Function" ...>
   <xsl:attribute name="type"> <xsl:text>Times</xsl:text> </xsl:attribute>
   <xsl:attribute name="resultType"> <xsl:text>double</xsl:text> </xsl:attribute>
   <xsl:element name="Operand" ...>
    <xsl:copy> <xsl:copy-of select="@*|node()"/> </xsl:copy>
   </xsl:element>
   <xsl:element name="Operand" ...>
   <xsl:element name="FloatScalar" ...> <xsl:text>1.27559</xsl:text> </xsl:element>
  </xsl:element>
 </xsl:element>
</xsl:template>
 ...
```

Figure 5. Example XSL Transformation

5.3 SLA mappings Document (SMD)

In this section, we present and discuss a sample SLA mapping document. Generally, SLA mappings can be defined using XSLT and XPath expressions.

Figure 5 shows a sample rule for XSL transformations where price defined in Euro is transformed to an equivalent price in US Dollars. Please note that for the case of simplicity we use a relatively simple example. Using XSLT even more complicated mappings can be defined, the explanation of which is out of the scope of this chapter.

As shown in Figure 5, the Euro metrics is mapped to the Dollar metric. In this example we define the mapping rule returning Dollars by using the *Times* function of *WSLA Specification* (see line 4). The *Times function* multiplies two operands: the first operand is the Dollar amount as selected in line 7, the second operand is the Dollar/Euro quote (1.27559) as specified in line 10. The dollar/euro quote can be retrieved by a Web service and is usually not hard coded.

With similar mapping rules users can map simple syntax values (values of some attributes etc.), but they can even define complex semantic mappings with considerable logic behind. Thus, even slightly different SLA templates can be translated into each other.

6. VieSLAF framework

In this section we present the architecture used for the semi-automatic management of *meta-negotiations* and *SLA mappings*. We discuss a sample architectural case study exemplifying the usage of *Vienna Service Level Agreement Framework - VieSLAF*.

As discussed in Section 3 *VieSLAF* framework represents the first prototype for the management of self-governing ICT Infrastructures. The *VieSLAF* framework enables application developers to efficiently develop adaptable service-oriented applications simplifying the handling with numerous Web service

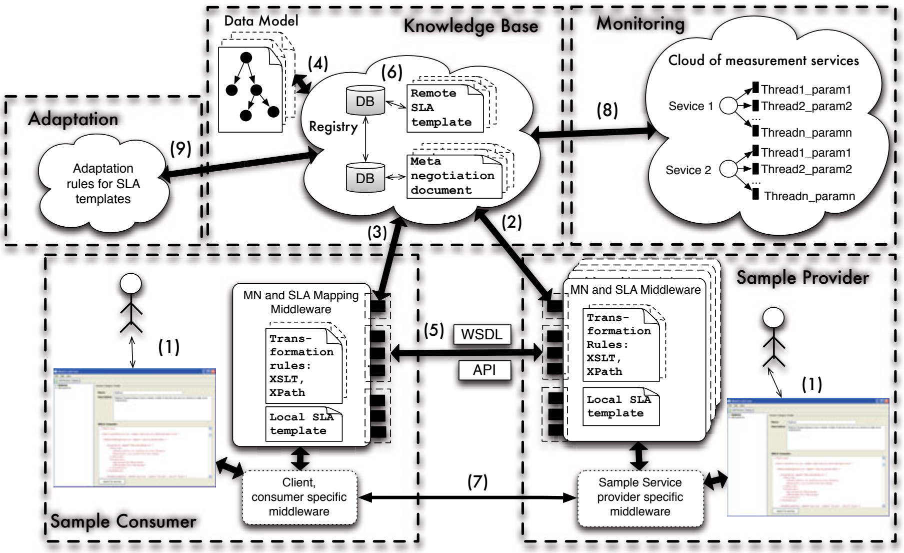

Figure 6. Extended VieSLAF Architecture with Monitoring and Taxonomies

specifications. The framework facilitates management of QoS models as for example management of meta-
negotiations [8] and SLA mappings [9]. Based on *VieSLAF* framework, a service provider may easily manage QoS models and SLA templates and frequently check whether selected services satisfy developer's needs e.g., specified QoS-parameters in SLAs. Furthermore, we discuss basic ideas about the adaptation of SLA templates.

We describe the *VieSLAF* components based on Figure 6. As shown in step (1) in Figure 6 users may access the registry using a GUI, browse through existing templates and meta-negotiation documents using the MN and SLA mapping middleware. In the next step (2), service provider specify MN documents and SLA mappings using the MN and SLA mapping middleware and submit it to the registry. Thereafter, in step (3), service consumer may query existing meta-negotiation documents, define own SLA mappings to remote templates and submit it to the registry. MN and SLA mapping middleware on both sides (provider's and consumer's) facilitates management of MNs and SLA mappings. Submitted MN documents and SLA mappings are parsed and mapped to a predefined data model (step 4). After meta-negotiation and preselection of services, service negotiation may start using the negotiation protocols, document languages, and security standards as specified in the MN document (step 5). During the negotiation SLA mappings and XSLT transformations are applied (step 6). After the negotiation, invocation of the service methods may start (step

7). SLA parameters are monitored using the monitoring service (step 8). Based on the submitted SLA mapping publicly available SLA templates are adapted reflecting the majority of local SLA templates (step 9).

7. Conclusion and Future Work

In this chapter we have presented the goals of the Foundations of Self-Governing ICT Infrastructures (FoSII) project and how these goals can be achieved using the principles of autonomic computing. We discussed novel meta-negotiation and SLA mapping solutions for Cloud services bridging the gap between current QoS models and Cloud middleware and representing important prerequisites for the establishment of autonomic Cloud services. We discussed the approaches for meta-negotiation and SLA mapping representing partial implementation of negotiation bootstrapping and service mediation approaches. Furthermore, we presented the *VieSLAF* framework used for the management of meta-negotiations and SLA mappings. Using *VieSLAF* service users can even monitor SLA parameters during the execution of the service calls. Finally, we discussed how SLA templates can be adapted based on the submitted SLA mappings.

As the next step of the FoSII project we plan to implement bootstrapping strategies where even consumer and provider, which understand different negotiation protocols and document languages can communicate with each other.

Acknowledgments

The work described in this chapter was partially supported by the Vienna Science and Technology Fund (WWTF) under grant agreement ICT08-018 Foundations of Self-governing ICT Infrastructures (FoSII).

References

[1] R.J. Al-Ali, O.F. Rana, D.W. Walker, S. Jha, and S. Sohail. G-qosm: Grid service discovery using qos properties. Computing and Informatics, 21:363–382, 2002.

[2] Amazon Simple Storage Services (S3), http://aws.amazon.com/s3/

[3] D. Ardagna, G. Giunta, N. Ingraffia, R. Mirandola, and B. Pernici. QoS-Driven Web Services Selection in Autonomic Grid Environments. Grid Computing, High Performance and Distributed Applications (GADA) 2006 International Conference, Montpellier, France, Oct 29 - Nov 3, 2006.

[4] A. P. Barros and M. Dumas. The Rise of Web Service Ecosystems. IT Professional 8(5):31–37, Sept./Oct., 2006.

[5] J. Blythe, E. Deelman, and Y. Gil. Automatically Composed Workflows for Grid Environments. IEEE Intelligent Systems 19(4):16–23, 2004.

[6] I. Brandic, S. Pllana, and S. Benkner. Specification, Planning, and Execution of QoS-aware Grid Workflows within the Amadeus Environment. Concurrency and Computation: Practice and Experience, 20(4):331–345, John Wiley & Sons, Inc., New Jersey, March 2008.

[7] I. Brandic, D. Music, S. Dustdar, S. Venugopal, and R. Buyya. Advanced QoS Methods for Grid Workflows Based on Meta-Negotiations and SLA-Mappings. The 3rd Workshop on Workflows in Support of Large-Scale Science. In conjunction with Supercomputing 2008, Austin, TX, USA, November 17, 2008.

[8] I. Brandic, S. Venugopal, Michael Mattess, and Rajkumar Buyya, Towards a Meta-Negotiation Architecture for SLA-Aware Grid Services. Technical Report, GRIDS-TR-2008-9, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Aug. 8, 2008.

[9] I. Brandic, D. Music, P. Leitner, and S. Dustdar. VieSLAF Framework: Increasing the Versatility of Grid QoS Models by Applying Semi-automatic SLA-Mappings. Vienna University of Technology, Technical Report, TUV-184-2009-02.pdf, 2008.

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and Ivona Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems, ISSN: 0167-739X, Elsevier Science, Amsterdam, The Netherlands, 2009, in press, accepted on Dec. 3, 2008.

[11] K. Czajkowski, I. Foster, C. Kesselman, V. Sander and S. Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh Scotland, July 2002.

[12] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef. Web services on demand: WSLA-driven automated management. IBM Systems Journal, 43(1), 2004.

[13] T. Glatard, J. Montagnat, and X. Pennec. A Probabilistic Model to Analyse Workflow Performance on Production Grids. 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), Lyon, France, pages 510-517, 19-22 May 2008.

[14] Google App Engine, http://code.google.com/appengine

[15] T. Guan, E. Zaluska, and D. De Roure. A Semantic Service Matching Middleware for Mobile Devices Discovering Grid Services. Advances in Grid and Pervasive Computing, Third International Conference, GPC 2008, Kunming, China, pages 422-433, May 25-28, 2008.

[16] Foundations of Self-Governing ICT Infrastructures (FoSII) Project, http://www.wwtf.at/projects/research_projects/details/index.php?PKEY=972_DE_O

[17] R. Jurca and B. Faltings. Reputation-based Service Level Agreements for Web Services. In Proceedings of 3rd International Conference on Service Oriented Computing, Amsterdam, The Netherlands, pages 396-409, December 12-15, 2005.

[18] J.O. Kephart and D.M. Chess, The vision of autonomic computing. Computer, 36(1):41–50, Jan 2003.

[19] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenges, IEEE Computer, 40(11): 64–71, November 2007

[20] A. Paschke, J. Dietrich, and K. Kuhla. A Logic Based SLA Management Framework. Semantic Web and Policy Workshop (SWPW), 4th Semantic Web Conference (ISWC 2005), Galway, Ireland, 2005.

[21] N. Oldham, K. Verma, A. P. Sheth, and F. Hakimpour. Semantic WS-agreement partner selection. Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006.

[22] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar. A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing. In Proceedings of the 2005 European Grid Computing Conference (EGC 2005), Amsterdam, The Netherlands, February, 2005.

[23] D. W. Walker, L. Huang, O. F. Rana, and Y. Huang. Dynamic service selection in workflows using performance data. Scientific Programming 15(4):235–247, 2007.
[24] Web Service Level Agreement (WSLA), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf