Including Energy Efficiency into Self-adaptable Cloud Services

Ivona Brandic, Vincent C. Emeakaroha, Michael Maurer, Schahram Dustdar
Distributed Systems Group, Vienna University of Technology, Vienna, Austria
{ivona,vincent,maurer,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, energy consumption should be minimized. In this paper 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.**

***Index Terms*—Cloud Computing; SLA management; autonomic computing;**

I. Introduction

Service-oriented Architectures (SOA) represent a promising approach for implementing ICT systems [1]. 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 [3]. 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 [2] or where massively scalable computing is made available to end users as a service as in case of *Cloud Computing* [4]. 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 [5]. Major goal of this paper 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. 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]. 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.

II. Overview

To facilitate dynamic, versatile, and adaptive IT infrastructures, SOA systems should react to environmental changes,

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

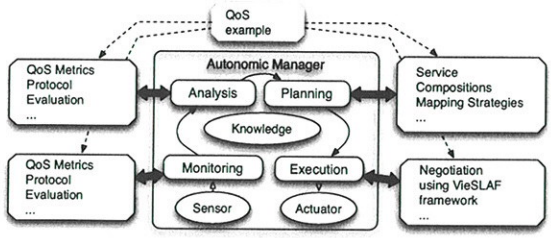

Fig. 1. General Architecture of an Autonomic System Explained on a QoS Example

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 [6], [7].

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 [8]. 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.
- **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.

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.

A. 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 *monitorig 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.

III. 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.

A. 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

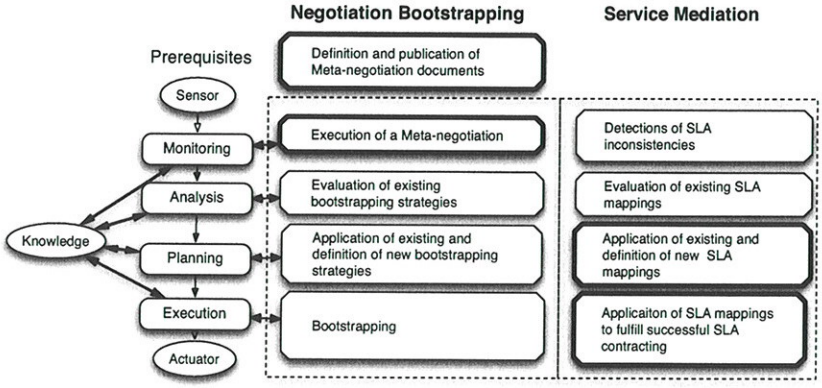

Fig. 2. Negotiation Bootstrapping and Service Mediation as Part of the Autonomic Process

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.

B. 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.

a) 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).

b) 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).

c) 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 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.

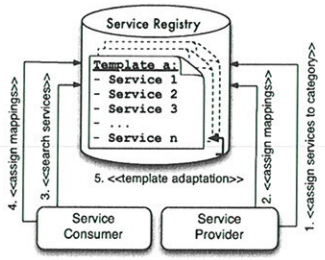

Fig. 4. Management of SLA-Mappings

IV. 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.

A. Management of SLA mappings

Figure IV-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 IV-A) and afterwards assign SLA mappings if

```
1. <meta-negotiation ...>
2.  <pre-requisite>
3.   <role name="consumer"/>
4.   <security> <authentication value="GSI" location="uri"/> </security>
5.   <negotiation-terms>
6.    <negotiation-term name="beginTime"/> <negotiation-term name="endTime"/>
8.   </negotiation-terms>
9.  </pre-requisite>
10. <negotiation>
11.  <document name="WSLA" value="uri" version="1.0"/>
12.  <protocol name="alternateOffers" schema="uri" version="1.0" location="uri"/>
13. </negotiation>
14. <agreement> <confirmation name="arbitrationService" value="uri"/> </agreement>
15.</meta-negotiation>
```

Fig. 3. Example Meta-negotiation Document

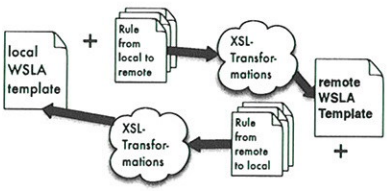

Fig. 5. Scenario for XSL Transformations

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.

B. Scenario for SLA mappings

Figure 5 depicts a scenario for defining XSL transformations. For the definition of SLA agreements we use Web Service Level Agreement (WSLA). 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 5 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.

ACKNOWLEDGMENT

The work described in this paper 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] A. P. Barros, M. Dumas. The Rise of Web Service Ecosystems. IT Professional 8(5): 31 – 37 , Sept.-Oct. 2006.

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

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

[4] 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.

[5] Foundations of Self-Governing ICT Infrastructures (FoSII) Project, http://www.infosys.tuwien.ac.at/linksites/FOSII

[6] J.O. Kephart, D.M. Chess, *The vision of autonomic computing*. Computer, 36:(1) pp. 41-50, Jan 2003.

[7] D. Ardagna, G. Giunta, N. Ingraffia, R. Mirandola and B. Pernici. QoS-Driven Web Services Selection in Autonomic Grid Environments. GADA 2006, International Conference, Montpellier, France, 2006.

[8] 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.

[9] I. Brandic, D. Music, Ph. Leitner, S. Dustdar. VieSLAF Framework: Enabling Adaptive and Versatile SLA-Management. Gecon09, In conjunction with Euro-Par 2009, 25- 28 August 2009, Delft, The Netherlands