Advanced QoS Methods for Grid Workflows Based on Meta-Negotiations and SLA-Mappings

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

Srikumar Venugopal, Rajkumar Buyya
Grid Computing and Distributed System (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{srikumar,raj}@csse.unimelb.edu.au

Abstract

In novel market-oriented resource sharing models, resource consumers pay for the resource usage and expect that non-functional requirements for the application execution, termed as Quality of Service (QoS), are satisfied. QoS is negotiated between two parties following the specific negotiation protocols and is recorded using Service Level Agreements (SLAs). However, most of the existing work assumes that the communication partners know about the SLA negotiation protocols and about the SLA templates before entering the negotiation. However, this is a contradictory assumption, if we consider computational Grids and novel, commercially oriented computing Clouds where consumers and providers meet each other dynamically. In this paper, we present novel meta-negotiation and SLA-mapping solutions for Grid workflows bridging the gap between current QoS models and Grid workflows, one of the most successful Grid programming paradigms. We illustrate the open research issues with a real world case study. Thereafter, we present document models for the specification of meta-negotiations and SLA-mappings. We discuss the architecture for the management of meta-negotiations and SLA-mappings as well as integration of the architecture into a Grid workflow management framework.

1 Introduction

In recent years, novel market-oriented resource sharing models have enhanced existing Grid computing concepts [21, 24, 4]. In market-oriented Grids, users pay for resource usage and therefore expect that requested *functional* as well as *non-functional* requirements of the application execution are fulfilled [1, 8, 20]. *Non-functional* requirements comprise application execution time, reliability, availability and similar issues. Non-functional requirements are termed as *Quality of Service (QoS)*, and are expressed and negotiated by means of *Service Level Agreements (SLAs)*. *Negotiation strategy* represents the internal decision making process used for resource selection. *Negotiation protocols* represents the publicly visible message exchange pattern during the negotiation process. *SLA templates* represent empty SLA documents with all required elements like parties, SLA parameters, metrics and objectives, but without QoS values [9]. SLAs are negotiated between two parties following the negotiation protocol, and are generated using the SLA templates available on both the consumers and providers sides. The necessity for appropriate market-oriented resource sharing models becomes even more evident, if we consider Grids as part of commercially used *Computing Clouds* [6].

There exists a large body of literature that deals with SLA-based Grid workflow negotiation and integration of QoS concepts into Grid workflow management tools [4, 19, 2]. However, most of these publications assume that the communication partner knows about the *SLA negotiation protocols* before entering the negotiation and that they have matching *SLA templates*. In commercially used Grids and especially in case of computational 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 docu-

ments 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.

In this paper, we approach the gap between existing QoS methods and Grid workflows by proposing an architecture for Grid workflow management with components for *meta-negotiations* and *SLA-mappings*. 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. Initial work on meta-negotiations for SLA-aware Grid services was presented in a previous publication [5]. 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.

The main contributions of this paper are therefore: (1) definition of *meta-negotiation* documents; (2) development of the strategies for the definition of *SLA-mapping* documents; (3) description of the Grid workflow architecture for the management of *meta-negotiations* and *SLA-mappings*; and (4) demonstration of *meta-negotiations* and *SLA-mappings* using a sample real-world Grid workflow.

The rest of this paper is organized as follows: Section 2 presents the related work. In Section 3, we present a real world case study used to illustrate meta-negotiations and SLA-mappings. In Section 4, we discuss meta-negotiations and Section 5 discusses the strategies for the SLA-mappings. Section 6 presents the architecture for the meta-negotiations and SLA-mappings, as well as integration of the architecture into a Grid workflow management tool. Section 7 presents our conclusions and describes the future work.

2 Related Work

Currently, a large body of work has been performed in the area of Grid workflow negotiation and QoS. Moreover, several projects are dealing with resource lookup based on functional and non-functional requirements. Ouelhadj et al. [17] discusse incorporation of SLA-based resource brokering into existing Grid systems. Wieczorek et al. [19] propose a novel approach for modeling scheduling problems as an extension of the multiple-choice knapsack problem. They present a general bi-criteria scheduling heuristic, based on dynamic programming, called the Dynamic Constraint Algorithm (DCA). Guo et al. [11] describe a web services based QoS-aware workflow management system (WfMS) utilized in context of GridCC project. Glatard et al. [10] discuss a probabilistic model of workflow execution time evaluated in context of EGEE grid infrastructure. Walker et. al [22] present an approach to dynamic workflow management and optimisation using near-realtime performance with strategies for choosing an optimal service, based on user-specified criteria, from several semantically equivalent Web services.

Venugopal et al. [21] propose a negotiation mechanism for advance resource reservation using the alternate offers protocol. Brandic et al. [4] present a holistic Grid infrastructure for specification, planing and execution of QoS-aware Grid workflows. In both these publications, it is assumed that each participating service understands the necessary negotiation protocol.

Al-Ali et al. [1] extend the service abstraction in the Open Grid Services Architecture (OGSA) for QoS properties, focusing on the application layer. A given service may indicate the QoS properties it can offer or it may search for other services based on specific QoS properties. Czajkowski et al. [8] propose generalized resource management model where resource interactions are mapped to a well-defined set of platform-independent SLAs, based on the Service Negotiation and Acquisition Protocol (SNAP). Condor's ClassAds mechanism is used to represent jobs, resources, and Condor daemons [18]. Dan et al. [9] present a framework for providing customers of Web services differentiated levels of service through the use of automated management and SLAs. Zhao et al. [25] discuss how semantic technologies can be used for workflow provenance, while Guan et al. [12] discuss their application for automatic location and selection of appropriate Grid services.

To the best of our knowledge, none of these approaches for the management of Grid workflows address *meta-negotiations (MN)* where participating parties may agree on a specific negotiation protocol, security standards or other negotiation pre-requisites, and *SLA-mappings* where participants may manage non-matching SLA templates. Furthermore, our approach ensures semantic matching of SLA templates between service provider and service consumer.

3 Case Study

In order to illustrate necessity for meta-negotiations and SLA-mappings in context of Grid workflows we use the sample workflow for maxillo facial surgery simulation (MFSS) that was introduced in a previous publication [3].

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

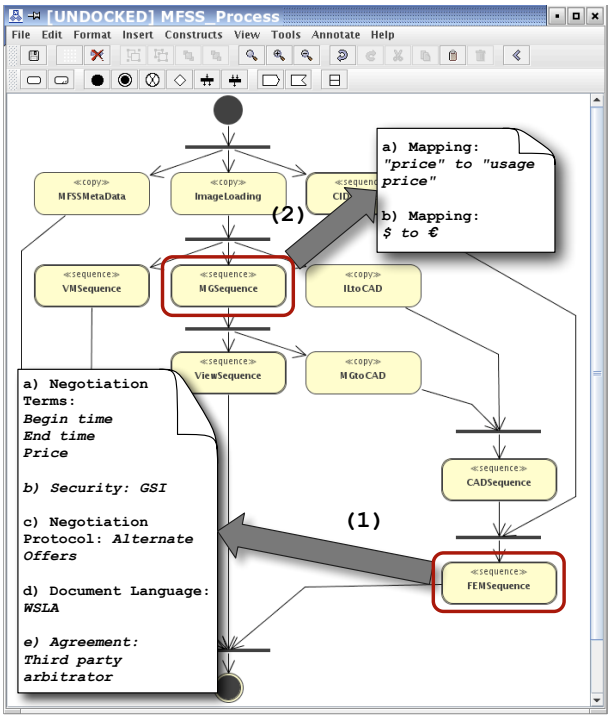

Figure 1. Sample MFSS Process

3.1 Maxillo Facial Surgery Simulation

The MFSS application facilitates the work of medical practitioners and provides the pre-operative virtual planning of maxillo-facial surgery. The application consists of a set of components which can run on a local machine or on different remote machines. As shown in Figure 1, these components may be organized as a Grid workflow in order to simplify the work of the end users. The main steps of the simulation are:

Mesh generation is used for the generation of meshes necessary for the finite element simulation. A sample complex mesh generation *MGSequence* activity is shown in the middle part of Figure 1.

Mesh manipulation defines the initial and boundary conditions for the simulation.

Finite Element Analysis is a fully parallel MPI application running on a remote HPC cluster. A sample complex finite element analysis *FEMSequence* activity is depicted in the down right corner of Figure 1.

3.2 Negotiation- and SLA-Restrictions

We model the MFSS workflow using *Amadeus*, a QoS-aware Grid modeling, planning, and execution tool [3]. Instead of specifying QoS with execution time and price constraints as presented previously [3], the user wants to express limitations on negotiation protocols and provided SLA templates as described in the following.

Meta-Negotiation. The restrictions on meta-negotiations are specified during the design time of the workflow. As depicted in Figure 1, part (1), meta-negotiation for the *MGSequence* activity (used for mesh generation) is specified by means of (a) negotiation terms, (b) security restrictions, (c) negotiation protocols, (d) document languages and (e) preconditions for the agreement establishment. *Negotiation terms* are specified as *begin time*, *end time*, and *price*. In order to initiate a negotiation, *GSI*[2] security is required. The negotiation is performed based on the alternate offers protocol. Therefore, the workflow application understands only the *alternate offers protocol*, and negotiation with resources which do not provide alternate offers protocol cannot be properly accomplished. Additional limitation considers document language used for the specification of SLAs. As shown in Figure 1, QoS is specified using *WSLA* [23]. Additionally, it is specified that for the successful agreement completion a *third party arbitration service* is required.

SLA-Mapping. The restrictions on SLA mappings are specified during the run-time of the workflow. We can assume that during the negotiation process, services that support the specified negotiation terms, negotiation protocols, and document language are found as described above. However, in our case study, the SLA template of the candidate service for finite element simulation, defined by sequence *FEMSequence*, varies from the workflow's SLA template. As shown in Figure 1, part (2), QoS constraint *price* is specified with the term *price*, whereas WSLA template of the candidate service contains QoS constraint termed *usage price*, see case (a). Furthermore, on the provider's side the price is charged in US Dollars, whereas workflow's QoS component expects Euros, see case (b).

In the following sections, we describe the methods and architecture for the specification and enforcement of *meta-negotiations* and *SLA-mappings*.

4 Grid Meta-Negotiations

In this section, we describe meta negotiation document based on the case study presented in Section 3. Thus, the meta-negotiation document depicted in Figure 3 represents the solution for the case study shown in Figure 1, part (1).

4.1 Meta-Negotiation Strategies

Usually, multiple meta-negotiation documents may be specified for a single workflow. Figure 2 shows different strategies for the specification of *meta-negotiation documents (MND)s*. A MND can be specified for a single task as shown in Figure 2, case (a). For the activities `A1`, `A4`

[2]Grid Security Infrastructure, http://www.globus.org/security/

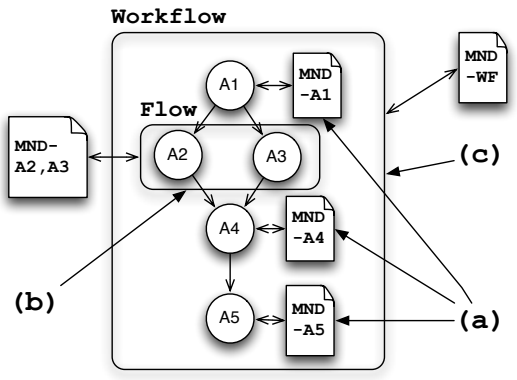

Figure 2. Attachment of MNM Policies

and A5 meta negotiation documents MND-A1, MND-A4, and MND-A5 are specified. For the complex activity flow and comprising activities A2 and A4 one MND is specified, namely MND-A2,A3 (see Figure, 2, case (b)). In the simplest case, the MND document is defined for the overall workflow as shown in case (c), Figure 2.

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. The document contains an <entity> element defining contact information, organization and ID of the participant. The <ID> element defines the unique identifier given to the meta-negotiation document by the registry. The publisher can update or delete the document using the identifier.

Furthermore, the <entity> element contains workflow specific information, such as the task ID that the MND is specified for. For example, in Figure 3, lines 7-9 specify that the MND is describing task nr. 5 of the specific workflow. Each meta-negotiation 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 are defined within the <pre-requisite> element (see Figure 3, lines 12–23). 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. For example, in Figure 3,

```
1.  <meta-negotiation
2.     xmlns:xsi="..."
3.     xsi:noNamespaceSchemaLocation="...">
4.     <entity>
5.       <contact name="..." .../>
6.       <ID name="1234"/>
7.       <workflow>
8.         <task>5</task>
9.       </workflow>
10.        ...
11.    </entity>
12.    <pre-requisite>
13.     <role name="consumer"/>
14.     <security>
15.       <authentication value="GSI"
16.         location="uri"/>
17.     </security>
18.     <negotiation-terms>
19.       <negotiation-term name="beginTime"/>
20.       <negotiation-term name="endTime"/>
21.       <negotiation-term name="price"/>
22.     </negotiation-terms>
23.   </pre-requisite>
24.   <negotiation>
25.     <document name="WSLA" value="uri"
26.       version="1.0"/>
27.     <protocol name="alternateOffers"
28.       schema="uri" version="1.0"
29.       location="uri"/>
30.   </negotiation>
31.   <agreement>
32.     <confirmation
33.       name="arbitrationService"
34.       value="uri"/>
35.   </agreement>
36. </meta-negotiation>
```

Figure 3. Example Meta-negotiation document

the consumer requires that the other party should be authenticated through the *Grid Security Infrastructure (GSI)* [13] (lines 15–16). 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*, *endTime*, and *price* (lines 19–21).

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 languages. Additional attributes specify the *URI* (Uniform Resource Indicator) to the API or WSDL for the

documents and their versions supported by the consumer (lines 25–26). 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 (lines 27–39).

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 lodged 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. For example, in Figure 3, a third party service, called "arbitrationService", is specified for confirming the agreement between the two parties.

The meta-negotiation architecture described here was experimentally evaluated and the results were presented in a previous publication [5].

5 SLA-Mappings

In this section, we describe sample documents for SLA mappings between inconsistent templates. SLA examples used here address the case study presented in Section 3, Figure 1, part (2). In the presented scenario, each WSLA template has to be published into a registry where the negotiation partners i.e., a provider and a consumer can find each other. The management of WSLA templates is described in Section 6. After publishing the WSLA (see Section 6) and after successful meta-negotiation (see Section 4), WSLA mappings can be defined as described in the following scenario.

5.1 Scenario for SLA-Mappings

Figure 4 depicts a scenario for defining XSL transformations.

WSLA templates are publicly available and published in a searchable registry. Each participant may download previously published WSLA templates and compare it 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 WSLA 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 WSLA template. During the negotiation process, the transformations is performed from the remote WSLA template to the local WSLA template and vice versa.

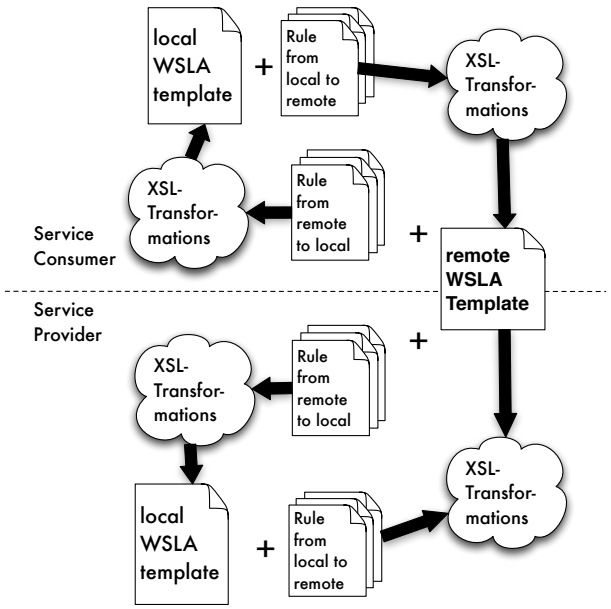

Figure 4. Scenario for XSL Transformations

The upper part of 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 WSLA template has to be translated into the local one. In that case, the remote WSLA plus the rules defining transformations from the remote to local WSLA deliver a WSLA which is complaint to the local WSLA. Thus, in this manner, the negotiation may be done using non-matching WSLAs.

As shown in the lower part of Figure 4, 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.

5.2 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 corresponding to the case study presented in Figure 2, part (2), case (a). The document is parsed for the SLAParameter *price* (see lines 2-6) and is replaced with the *usage price* parameter as shown in line 8.

A similar document can be defined for the transformations between Dollars and Euros as described in the case study (see Section 3). In this case, we define the mapping rule which selects the calculation function of the remote WSLA and returns result in Euro by multiplying the calcu-

```
1.   ...
2.   <xsl:template match=""/ns:SLA/
3.    ns:ServiceDefinition/
4.    ns:WSDLSOAPOperation/
5.    ns:SLAParameter/
6.    @name[.='price']"">
7.    <xsl:attribute name='{name()}'>
8.       <xsl:text>usage price</xsl:text>
9.    </xsl:attribute>
10. </xsl:template>
11. ...
```

Figure 5. Example XSL Transformation

lated value with Euro/Dollar factor. With similar mapping rules, users can not only map simple syntax values (values of some attributes etc.), they can even define complex semantic mappings backed by complex logic.

6 Architecture

In this section, we present the architecture used for the management of the *meta-negotiation documents* and *SLA-mappings* (MNSM) in the context of a QoS-aware Grid workflow system. First, we discuss the architectural components for the management of *meta-negotiations* and *SLA-mappings*. We describe the components in detail and give a sample architectural case study. Finally, we incorporate the *meta-negotiations* and *SLA-mappings* components into a QoS-aware Grid workflow architecture and discuss it based on the introduced case study.

6.1 Meta-Negotiation and SLA-Mapping (MNSM) Architecture

6.1.1 Meta-Negotiation Infrastructure

In order to create a case study that tests the proposed meta-negotiation framework in practice, we have extended a previous publication on negotiation of advance reservations using the alternate offers protocol [21] to incorporate the meta-negotiation framework. The architecture followed in this case study is shown in Figure 6. It consists of the registry for meta-negotiation documents and the meta-negotiation middleware on both the provider and consumer sides.

In our architecture, the service provider role is carried out by Aneka [7], which is a resource management system for enterprise Grids composed of machines running Microsoft Windows operating system. Aneka provides facilities for advance reservation of computing nodes and supports flexible scheduling of applications constructed using different parallel programming models such as bag-of-tasks and dataflow computing. The Gridbus Broker [20] maps jobs to appropriate resources considering various restrictions specified by terms of *functional* and *non-functional* requirements. *Functional requirements* include but are not limited to task and data dependencies such as, for example, a sequence of tasks is required to execute a specific application. *Non-functional requirements* include QoS parameters such as budget restrictions, and a deadline for execution. The broker can guarantee the end-user's deadline requirement only if it is able to reserve nodes on resources in advance. Therefore, in this respect, the broker functions as a consumer that requests reservations from the provider.

6.1.2 SLA-Mappings Infrastructure

In order to create a case study that tests the proposed SLA-mapping framework in practice, we developed novel infrastructure as depicted in Figure 6 based on the *VRESCo* framework [14]. The *VRESCo* framework enables application developers to efficiently develop service-oriented application simplifying the handling with numerous Web service specifications such as UDDI. The framework facilitates dynamic service management by utilizing client-side libraries which transparently handle service communication as well as publishing, searching, querying and composing of services. In *VRESCo*, services are bound based on dynamic proxies, which frequently check whether selected services satisfy developer's needs e.g., specified QoS-parameters in SLAs. We extended the *VRESCo* framework with the features for the management of the WSLA-templates and SLA-mappings as described in the following.

Using the SLA-mapping features of *VRESCo*, users may search for service templates, publish services and define SLA-mappings to existing services. We classify service templates into categories i.e., for each specific domain, as for example medical, telecommunication or financial domain we provide a single template. Using an appropriate GUI, which is subject of ongoing research, users may browse through templates, publish services to the registry and define SLA-mappings from local WSLAs to remote WSLAs and vice versa. As depicted in Figure 6, SLA-templates are mapped to *VRESCo's* predefined data model based on taxonomies [15].

6.1.3 Registry

We implemented two distinct registries for meta-negotiations and SLA-mappings, described as follows:

Meta-negotiation registry. The registry is a searchable repository for meta-negotiation documents that are created by the participants. Currently, this is implemented as a PostgreSQL database with a web service front end that provides

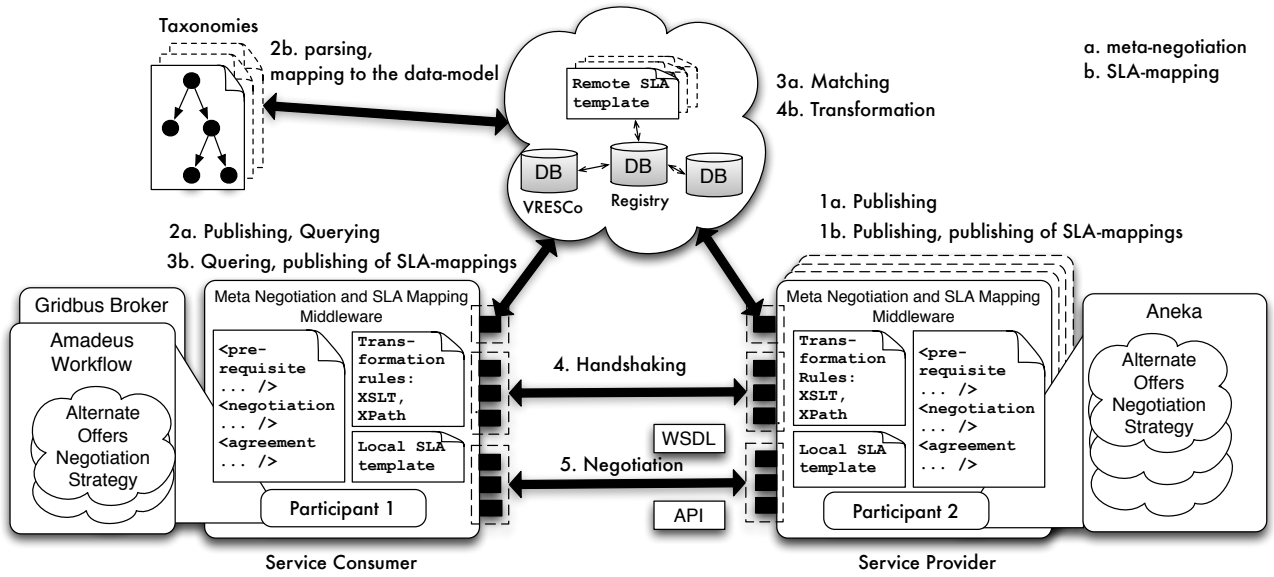

Figure 6. Architecture for meta-negotiations and SLA-mappings in heterogeneous Grids with sample provider and consumer

the interface shown in Figure 7. However, it is possible to host the registry using a cloud of databases hosted on a service provider such as Google App Engine[3] or Amazon S3. When a meta-negotiation document is published, the registry assigns it a unique identifier (ID) that can then be used for subsequent operations. The query call tries to find the documents that match the maximum number of attributes of the search query. It returns an array of IDs of these documents to the caller which can then fetch each one through the `getDocument` call. Methods depicted in Figure 7 in lines 1-4 depict the methods for the manipulation of MNDs, whereas methods depicted in lines 5-9 represent the methods for the manipulation of SLA-mappings.

```
1. publish(XMLdocument);
2. update(XMLdocument);
3. query(XMLdocument);
4. getDocument(ID);
5. createTemplateCategory(TemCategory);
6. createAttributeMapping(ProviderAttrMapp);
7. createAttributeMapping(ConsumerAttrMapp);
8. createService(Service);
9. findServices(ConsumerServiceRequest);
```

Figure 7. Registry Methods

SLA-mapping registry. We used MS-SQL 2008 for the SLA-mappings database. The database is manipulated based on the role-model. The registry methods are implemented as *WCF*[4] *services* and can be accessed with appropriate access rights only. We define three roles: *service consumer*, *service provider* and *registry administrator*. *Service consumer* is able to search suitable services for the selected service categories e.g., by using the method *findServices*, as depicted in Figure 7, line 9. Service consumer may also create *SLA-mappings* as depicted in line 7. *Service provider* may publish his services and bind it to a specific template category using the method *createService*. Furthermore, he may define *SLA-mappings* by using the method *createAttributeMapping*. Registry administrator may create, update and delete service categories. Please note that for all *create* methods depicted in lines 5-8, we have implemented the corresponding CRUD methods. Each *template category* is identified with an unique name and is stored in the database as an XML document. Each service is identified with a name and is described with a *WSDL-URI* and filled *SLA-template* of the selected category. For each service, multiple SLA-mapping documents may be defined.

6.1.4 Meta-Negotiation Middleware

The *meta-negotiation middleware* facilitates the publishing of the meta-negotiation documents into the registry and the integration of the meta-negotiation framework into the ex-

[3] http://code.google.com/appengine

[4] Windows Communication Foundation

isting clients (e.g. workflow tools) and/or service infrastructure, including, for example, negotiation or security clients. Besides acting as a client for publishing and querying meta-negotiation documents (steps 1a and 2a in Figure 6), the middleware delivers necessary information for the existing negotiation clients i.e., information for the establishment of the negotiation sessions (step 4, Figure 6) and information necessary to start a negotiation (step 5 in Figure 6). As shown in Figure 6, each service consumer may negotiate with multiple service providers concurrently. As mentioned in Section 4, even the reverse may happen as well, wherein a consumer advertises a job. In such cases, the providers would negotiate with multiple consumers.

After querying the registry and applying a client-based strategy for the selection of the appropriate service, the information from the service's meta-negotiation document is parsed. Thereafter, meta-negotiation information is incorporated into the existing client software using a dependency injection framework such as Spring[5]. This dependency injection follows an Inversion of Control approach wherein the software is configured at runtime to invoke services that are discovered dynamically rather than known and referenced beforehand. This is suitable in the context of meta-negotiation wherein a participant discovers others at runtime through the registry and has to dynamically adapt based on the interfaces provided by his counterpart (usually through a WSDL document).

On the consumer side, the middleware queries the registry and obtains matching meta-negotiation documents. The middleware parses the meta-negotiation document of the selected provider and dynamically injects the interfaces discovered from the WSDLs in the document for security, negotiation and arbitration services into the existing abstract clients. Currently, we support semi-automatic integration of existing clients into meta-negotiation middleware wherein the existing clients are extended with the XML-based configuration files which are then automatically populated with the discovered interfaces.

6.1.5 SLA-Mapping Middleware

As already mentioned in Section 6.1.3, SLA-mapping middleware is based on a bunch of WCF services. For the sake of brevity, in the following we discuss just a few of them. *RegistryAdministrationService* provides methods for the manipulation of the database where administrator rights are required e.g., creation of template categories. Another example represents *WSLAMappingService* which is used for the management of SLA mappings by service consumer and service provider. *WSLAQueryingService* is used to query SLA mapping database. Database can be queried based on template category, SLA attributes and similar attributes.

Before publishing a service by a provider, the submitted SLA-template is parsed by a WSLA parser as depicted by step 1b in Figure 6. Within the SLA-mapping framework, it is not prescribed which parser to use, moreover we prescribe the interface which has to be implemented by each parser. For the creation of the concrete instance of the parser, we use the *abstract factory* pattern. Currently, we have implemented a version with the recent Language Integrated Query (LINQ) technology from .NET 3.5. Each *WSLA parser* takes as input an *SLA template*, parses this document in *SLA types* with *SLA elements* and *SLA attributes*, and maps it to the predefined data model, as defined by step 2b in Figure 6.

Service consumers may now search for appropriate services through *WSLAQueryingService* and define appropriate *SLA-mappings* by using the method *createAttributeMapping* as depicted in step 3b. Thereafter, service consumer may send request with a filled SLA template to the registry using the *WSLAQueryingService*. Initially the existence of prior SLA mappings defined for that specific service consumer is checked. If so, these will be used for transformation using registry's internal *TransformatorService* as depicted in in step 4b. TransformatorService is implemented using an *AbstractFactory* pattern, thus different transformation services can be easily incorporated into the SLA mapping middleware. After the transformation, service negotiation may start as depicted in steps 4 and 5.

6.2 Workflow Management

Figure 8 shows the integration of the meta-negotiation and SLA-mapping architecture, presented in Section 6.1, into *Amadeus*, a Grid workflow tool [4]. Workflow modeling is done using the *Teuta* tool where QoS constraints, meta-negotiation documents and SLA-mappings can be specified using UML standard. QoS constraints, meta-negotiation documents, and SLA-mappings are specified in a visual way by selecting the appropriate task and filling up the property panel of the Teuta tool. The UML representation is translated into the XML representation following the syntax of Quality of Service aware Grid Workflow Language (QoWL) [3]. QoWL excerpt of the MFSS workflow (see Figure 1, part (1)) is depicted in Figure 9.

As depicted in lines 1-12, the QoWL element represents a *sequence* activity. Specification of QoS constraints is done using `<qos-constraints>` element (see lines 3-11). The *reqDescVar* attribute specifies the request descriptor with the meta-data necessary for the performance prediction. The *mnd* attribute specifies the path to the meta-negotiation document depicted in Figure 3, and the *SLA-mapp* attribute specifies the path to the SLA-mapping doc-

[5]http://www.springframework.org

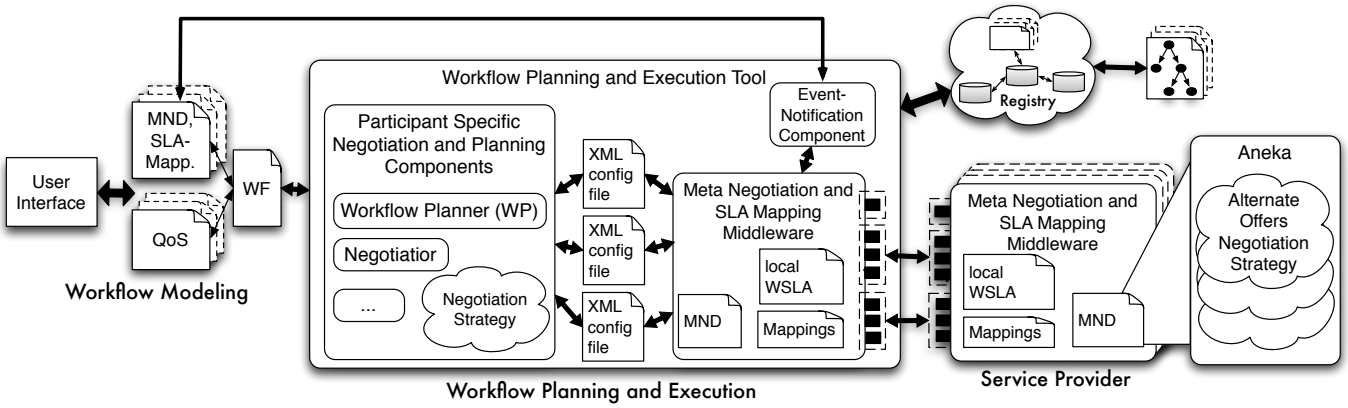

Figure 8. Workflow Management Tool with Components for Meta-Negotiations and SLA-Mappings

```
1.  <sequence name="MGSequence" ... >
2.    ...
3.    <qos-constraints reqDescVar="..."
4.      mnd="..." SLA-mapp="...">
5.      <qos-constraint name="beginTime"
6.        value="..."/>
7.      <qos-constraint name="endTime"
8.        value="..."/>
9.      <qos-constraint name="price"
10.       value="..." />
11.   </qos-constraints>
12. </sequence>
```

Figure 9. QoWL example

ument depicted in Figure 5. QoS constraints are defined by the element `<qos-constraint>`.

After translation of the workflow UML representation into QoWL, meta negotiation is conducted as described in the meta-negotiation scenario, Figure 6, Section 6.1.4. Meta-negotiation and SLA-mapping middleware is integrated into the *Amadeus* tool using XML descriptors and following the concepts of Inversion of Control. As shown in Figure 8, XML configuration files bridge the MNSM with the participant specific negotiation and planning components. After the meta-negotiation, negotiations with selected resources may start as described in [4].

Currently, we are developing an infrastructure for the detection of inconsistent WSLA templates using event-based notification mechanisms. Thus, negotiation participants who want to negotiate even with services which have non-matching SLAs may subscribe for specific topic e.g., terms of negotiation. If services that require SLA-mappings are discovered during the meta negotiation process e.g., due to inconsistent terms of negotiation, all subscribed participants are notified. Thereafter, SLA-mappings may be defined using the appropriate GUI and stored in the database. Finally, negotiation between services with non-matching SLAs may start.

7 Conclusion and Future Work

In this paper, we have presented advanced QoS methods for meta-negotiations and SLA-mappings in Grid workflows. We exemplified our research issues with a real-world case study with Grid workflows for maxillo facial surgery simulation. Thereafter, we discussed the meta-negotiation documents through which each participant may state supported protocols, and document languages as well as the pre-requisites for starting negotiations and establishing agreements. We presented novel methods for SLA-mappings which enables negotiation between partners with non-matching SLA templates. Furthermore, since all SLA templates have to be mapped to a data model, we eliminated semantic inconsistencies between consumer's and provider's SLA templates. We developed an architecture for the management of meta-negotiation documents and SLA-mappings, and incorporated that architecture into a Grid workflow management tool. In the future we plan to test our approach with real-life Grid applications and develop strategies to handle negotiations across incompatible protocols and documents.

Acknowledgments

The work described in this paper was supported by the the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement 215483 (S-Cube) and by the Australian Research Council and the Dept. of Innovation, Industry and Scientific Research under Discovery Project and International Science Linkage grants respectively.

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

[3] I. Brandic, S. Pllana, S. Benkner. High-level Composition of QoS-aware Grid Workflows. *An Approach that Considers Location Affinity*. Workshop on Workflows in Support of Large-Scale Science. In conjunction with the 15th IEEE International Symposium on High Performance Distributed Computing, Paris, France, June 2006.

[4] I. Brandic, S. Pllana, 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.

[5] I. Brandic, S. Venugopal, Michael Mattess, and R. 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.

[6] R. Buyya, C. S. Yeo, and S. Venugopal, *Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities*. 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), Sept. 25-27, 2008, Dalian, China.

[7] X. Chu, K. Nadiminti, Ch. Jin, S. Venugopal, and R. Buyya *Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications*. Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (e-Science 2007), Dec. 10-13, 2007, Bangalore, India.

[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] 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.

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

[11] L. Guo, A. S. McGough, A. Akram, D. Colling, J. Martyniak, M. Krznaric. *Enabling QoS for Service-Oriented Workflow on GRID*. Seventh International Conference on Computer and Information Technology (CIT 2007), pp. 1077-1082, Fukushima, Japan, October 16-19, 2007.

[12] T. Guan, E. Zaluska, 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, pp. 422-433 Kunming, China, May 25-28, 2008.

[13] I. Foster, and C. Kesselman, and G. Tsudik, and S. Tuecke. *A Security Architecture for Computational Grids*, Proc. 5th ACM Conference on Computer and Communications Security Conference, San Francisco, CA, USA, ACM Press, New York, USA, 1998.

[14] A. Michlmayr, F. Rosenberg, Ch. Platzer, M. Treiber, S. Dustdar. *Towards Recovering the broken SOA Triangle - A Software Engineering Perspective*. In Proceedings of the 2nd International Workshop on Service-oriented Software Engineering (IW-SOSWE'07), Dubrovnik, Croatia, September 2007.

[15] F. Rosenberg, P. Leitner, A. Michlmayr, S. Dustdar. *Integrated Metadata Support for Web Service Runtimes*. Proceedings of the Middleware for Web Services Workshop (MWS'08), co-located with the 12th IEEE International Distributed Object Computing Conference (EDOC'08), 15-19. September 2008, Munich, Germany.

[16] I. J. Taylor, E. Deelman, D. B. Gannon. *Workflows for e-Science*. Springer Verlag, 2005.

[17] 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.

[18] D. Thain, T. Tannenbaum, and M. Livny. *Distributed Computing in Practice: The Condor Experience*. Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

[19] M. Wieczorek, S. Podlipnig, R. Prodan, T Fahringer. *Bicriteria Scheduling of Scientific Workflows for the Grid Cluster Computing and the Grid*. IEEE International Symposium on Cluster Computing and the Grid, Lyon, France, 19-22 May 2008.

[20] S. Venugopal, R. Buyya and L. Winton, *A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids*, Concurrency and Computation: Practice and Experience, 18(6): 685-699, Wiley Press, New York, USA, May 2006.

[21] S. Venugopal, X. Chu, R. Buyya. *A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol*. 16th International Workshop on Quality of Service (IWQoS 2008), June 2-4, 2008, University of Twente, Enschede, The Netherlands.

[22] D. W. Walker, L. Huang, O. F. Rana, Y. Huang. *Dynamic service selection in workflows using performance data*. Scientific Programming 15(4): 235-247 (2007).

[23] Web Service Level Agreement (WSLA), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[24] J. Yu, R. Buyya. *A Taxonomy of Workflow Management Systems for Grid Computing*. Journal of Grid Computing, 3(3-4):171-200, Springer Science + Business Media B.V., New York, USA, September 2005.

[25] J. Zhao, C. Goble, R. Stevens, D.Turi. *Mining Taverna's semantic web of provenance* Concurrency and Computation: Practice & Experience 20(5), John Wiley & Sons, Inc., New Jersey, April 2008.