An SLA-based Resource Virtualization Approach For On-demand Service Provision

Attila Kertesz
MTA SZTAKI
1518 Budapest, Hungary
P.O. Box 63.
attila.kertesz@sztaki.hu

Gabor Kecskemeti
MTA SZTAKI
1518 Budapest, Hungary
P.O. Box 63.
kecskemeti@sztaki.hu

Ivona Brandic
TU Vienna
1040 Vienna, Austria
Argentinierstr. 8/181-1
ivona@infosys.tuwien.ac.at

ABSTRACT

Cloud computing is a newly emerged research infrastructure that builds on the latest achievements of diverse research areas, such as Grid computing, Service-oriented computing, business processes and virtualization. In this paper we present an architecture for SLA-based resource virtualization that provides an extensive solution for executing user applications in Clouds. This work represents the first attempt to combine SLA-based resource negotiations with virtualized resources in terms of on-demand service provision resulting in a holistic virtualization approach. The architecture description focuses on three topics: agreement negotiation, service brokering and deployment using virtualization. The contribution is also demonstrated with a real-world case study.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks, Distributed Systems

General Terms

Design, Languages, Management, Experimentation

Keywords

SLA negotiation, Resource virtualization, Service Brokering, On-demand deployment

1. INTRODUCTION

Grid Computing [16] has succeeded in establishing production Grids serving various user communities all around the world. The emerging Web technologies have already affected Grid development; the latest solutions from related research fields (e.g. autonomous computing, P2P, etc.) also need to be considered in order to successfully transform the currently separated production Grids and Service oriented Architectures to the Internet of Services [25]. Cloud Computing [5] is a novel candidate that aims at creating this synergy; therefore we consider a cloud-like architecture focusing on agreement negotiation, service brokering and deployment using advanced virtualization techniques. Both Grids and Service Based Applications (SBAs) already provide solutions for executing complex user tasks. The web service model is based on three actors: a service provider, a service requester and a service broker [29]. Solutions building on this model use well-established and widely used technologies [29] that enable the collaboration of these three parties to fulfill service executions required by users. The newly emerging demands of users and researchers call for expanding this service model with business-oriented utilization (agreement handling) and support for human-provided and computation-intensive Grid services. Most of related works consider either virtualization approaches [12] [24] [18] without taking care of Service-Level Agreements (SLAs) or concentrates on SLA management neglecting the appropriate resource virtualizations [27] [6].

In this paper we propose a novel holistic architecture considering resource provision using the virtualization approach and combining it with the business-oriented utilization used for the SLA agreement handling. Thus, we provide an integrative infrastructure for on demand service provision based on SLAs. The main contributions of this paper include: (i) presentation of the novel architecture for the SLA-based resource virtualization and on-demand service provision, (ii) description of the architecture including *meta-negotiation, meta-brokering, brokering and automatic service deployment* and (iii) *demonstration* of the presented approach based on a case study. In the following section we summarize related works, in Section 3 we introduce the overall architecture and define the participating components and list the general utilization steps. In Section 4 the components of the three problem areas are detailed, and Section 5 presents a case study using the architecture. Finally Section 6 concludes the paper.

2. RELATED WORK

Though cloud-based service execution is rarely studied yet, some related works have already started to investigate, how business needs and more dynamicity could be represented in the web service model. The envisioned framework in [10] proposes a solution to extend this model by introducing and using semantic web services. The need for SLA handling, brokering and deployment also appears in this vision, but they focus on using ontology and knowledge-based

VTDC'09, June 15, 2009, Barcelona, Spain.

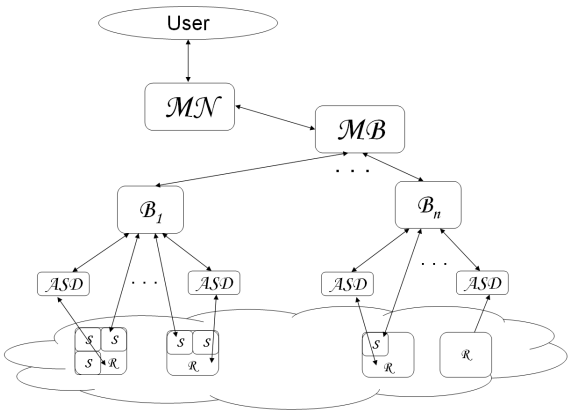

Figure 1: SRV architecture.

approaches. Works presented in [23] [21] discusses incorporation of SLA-based resource brokering into existing Grid systems. The Rudder framework [19] facilitates automatic Grid service composition based on semantic service discovery and space based computing. Venugopal et al. propose a negotiation mechanism for advance resource reservation using the alternate offers protocol [31], however, it is assumed that both partners understand the alternate offers protocol.

Regarding meta-brokering, LA Grid [26] developers aim at supporting grid applications with resources located and managed in different domains. They define broker instances, each of them collects resource information from its neighbors and save the information in its resource repository. The Koala grid scheduler [11] was redesigned to inter-connect different grid domains. They use a so-called delegated matchmaking (DMM), where Koala instances delegate resource information in a peer-2-peer manner. Gridway introduced a Scheduling Architectures Taxonomy [30], where Gridway instances can communicate and interact through grid gateways. These instances can access resources belonging to different Grid domains. Comparing the previous approaches, we can see that all of them use high level brokering that delegate resource information among different domains, broker instances or gateways. These solutions are almost exclusively used in Grids, they cannot co-operate with different brokers operating in pure service-based or cloud infrastructures. On the contrary, our proposed Meta-Broker can manage diverse, separated brokers.

Finally service deployment solutions are focusing on the deployment process itself and do not leverage their benefits on higher level. For example the Workspace Service (WS) [12] as a Globus incubator project supports wide range of scenarios involving virtual workspaces, virtual clusters and service deployment from installing a large service stack like ATLAS to deploy a single WSRF service if the Virtual Machine (VM) image of the service is available. The WS is designed to support several virtual machines – XEN [3], VMWare, VServer – to accomplish its task. Then the XenoServer open platform [24] is an open distributed architecture based on the XEN virtualization technique. It is aiming at global public computing. The platform provides services for server lookup, registry, distributed storage and a widely available virtualization server. Also the VMPlants [18] project proposes an automated virtual machine configuration and creation service which is heavily dependent on software dependency graphs. This project stays within cluster boundaries.

3. SLA-BASED RESOURCE VIRTUALIZATION (SRV) ARCHITECTURE

In this paper we present a unified service architecture that builds on three main areas: agreement negotiation, brokering and service deployment using virtualization. We suppose that service providers and service consumers meet on demand and usually do not know about the negotiation protocols, document languages or required infrastructure of the potential partners. First we introduce the general architecture naming the novelties and open issues, then we detail the aforementioned three main areas with respect to the shown architecture. Figure 1 shows our proposed, general architecture. Next we define the actors of the architecture:

- User: A person, who wants to use a service
- MN – Meta-Negotiator: A component that manages Service-level agreements. It mediates between the user and the Meta-Broker, selects appropriate protocols for agreements; negotiates SLA creation, handles fulfillment and violation.
- MB – Meta-Broker: Its role is to select a broker that is capable of deploying a service with the specified user requirements.
- B – Broker: It interacts with virtual or physical resources, and in case the required service needs to be deployed it interacts directly with the ASD.
- ASD – Automatic Service Deployment: It installs the required service on the selected resource on demand.
- S – Service: The service that users want to deploy and/or execute.
- R – Resource: Physical machines, on which virtual machines can be deployed/installed.

The following detailed steps are done during utilization with respect to the steps shown in Figure 2:

- User starts a negotiation for executing a service with certain QoS requirements (specified in a Service Description (SD) with an SLA) (step 1)
- MN asks MB, if it could execute the service with the specified requirements (step 2)
- MB matches the requirements to the properties of the available brokers and replies with an acceptance or a different offer for renegotiation (step 3)
- MN replies with the answer of MB. Steps 1-4 may continue for renegotiations until both sides agree on the terms (to be written to an SLA document)
- User calls the service with the SD and SLA (step 5)
- MN passes SD and the possibly transformed SLA (to the protocol the selected broker understands) to the MB (step 6)

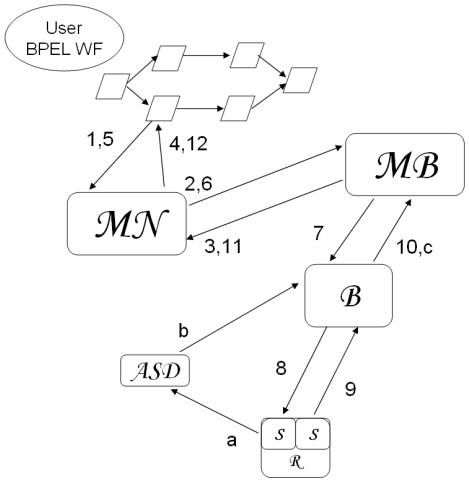

Figure 2: Detailed steps during SRV utilization.

- MB calls the selected Broker with SLA and a possibly translated SD (to the language of the Broker) (step 7)
- The Broker executes the service with respect to the terms of the SLA (step 8)
- In steps 9, 10, 11 and 12, the result of the execution is reported to the Broker, the MB, the MN, finally to the User (or workflow engine)
- ASD monitors the states of the virtual resources and deployed services (step a)
- ASD reports service availability and properties to its Broker (step b)
- All Brokers report available service properties to the MB (step c)

The previously presented sample architecture and the detailed utilization steps show that agreement negotiation, brokering and service deployment are closely related and each of them requires extended capabilities in order to interoperate smoothly. Nevertheless each part represents an open issue, since agreement negotiation, SLA-based service brokering and on-demand adaptive service deployment are not supported by current solutions in cloud-like environments.

4. REQUIREMENTS AND SOLUTIONS TO REALIZE SRV

In this section we detail three main categories, where the basic requirements of SRV-like systems arise. We place these areas in the SRV architecture shown in Figure 1, and detail the related parts of the proposed solution. We also emphasize the interactions among these components in order to build one coherent system.

In our proposed approach users describe the requirements for an SLA negotiation on a high level using the concept of meta-negotiations. During the meta-negotiation only those services are selected, which understand specific SLA document language and negotiation strategy or provide a specific security infrastructure. After the meta-negotiation process, a meta-broker selects a broker that is capable of deploying a service with the specified user requirements. Thereafter, the selected broker negotiates with virtual or physical resources using the requested SLA document language and using the specified negotiation strategy. Once the SLA negotiation is concluded, service can be deployed on the selected resource using the virtualization approach.

4.1 Agreement negotiation

As shown in Figure 1 to realize such a system we need to provide the following means of negotiation:

- User – MN: the User supplies a general meta-negotiation document
- MN – MB: they agree on specific negotiation documents containing specific negotiation strategy to be used, negotiation protocols to be used (WSLA, WS-Ag,) , terms of negotiation (e.g. time, price,), security infrastructure to be used
- MB – B: they agree on a specific SLA written in a specific SLA language (e.g. WSLA, WS-Agreement) containing concrete SLA parameters like concrete execution time, concrete price, etc.
- B – ASD: they agree on a specific service to be available on the ASD managed resources with the resource constraints resulted from the higher level negotiation – the service is going to be able to use the requested resources without disruptions from other parties
- Furthermore we need on each level (MN, MB, B, ASD) a negotiator which is responsible for generating and interpreting SLAs.

Before committing themselves to an SLA, the user and the provider may enter into negotiations that determine the definition and measurement of user QoS parameters, and the rewards and penalties for meeting and violating them respectively. The term negotiation strategy represents the logic used by a partner to decide which provider or consumer satisfies his needs best. A negotiation protocol represents the exchange of messages during the negotiation process. Recently, many researchers have proposed different protocols and strategies for SLA negotiation in Grids [22]. However, these not only assume that the parties to the negotiation understand a common protocol but also assume that they share a common perception about the goods or services under negotiation. In reality however, a participant may prefer to negotiate using certain protocols for which it has developed better strategies, over others. Thus, the parties to a negotiation may not share the same understanding that is assumed by the earlier publications in this space.

In order to bridge the gap between different negotiation protocols and scenarios, we propose a so-called meta-negotiation architecture [4]. *Meta-negotiation* is needed by means of a meta-negotiation document where participating parties may express: the pre-requisites to be satisfied for a negotiation, for example a specific authentication method required or terms they want to negotiate on (e.g. time, price, reliability); the negotiation protocols and document languages for the specification of SLAs that they support; and conditions for the establishment of an agreement, for example, a

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

Figure 3: Example Meta Negotiation Document

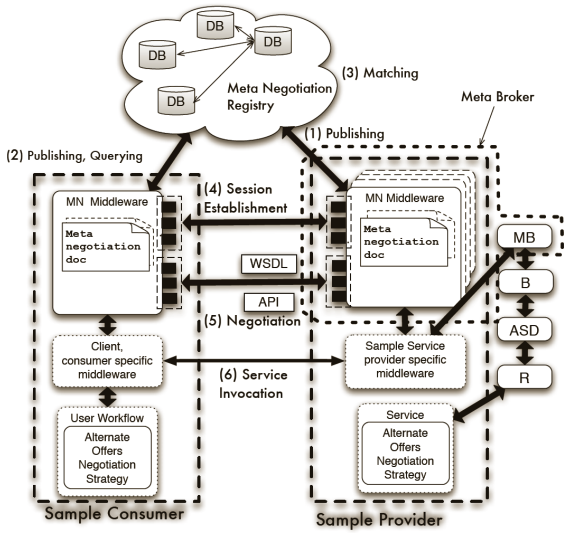

Figure 4: Meta-negotiation in SRV.

required third-party arbitrator. These documents are published into a searchable registry through which participants can discover suitable partners for conducting negotiations. In our approach, the participating parties publish only the protocols and terms while keeping negotiation strategies hidden from potential partners.

The participants publishing into the registry follow a common document structure that makes it easy to discover matching documents (as shown in Figure 3). This document structure 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 a unique ID of the participant. Each meta-negotiation comprises three distinguishing parts, namely *pre-requisites, negotiation and agreement* as described in the following paragraph.

As shown in Figure 3 prerequisites define the role a participating party takes in a negotiation, the security credentials and the negotiation terms. For example, the security element specifies the authentication and authorization mechanisms that the party wants to apply before starting the negotiation process. For example, the consumer requires that the other party should be authenticated through the Grid Security Infrastructure (GSI) [7]. The negotiation terms specify QoS attributes that a party is willing to negotiate and are specified in the <negotiation-term> element. As an example, the negotiation terms of the consumer are beginTime, endTime, and price. Details about the negotiation process are defined within the <negotiation> element. Each document language is specified within <document> element. 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. Figure 4 emphasizes a meta-negotiation infrastructure embedded into the agreement negotiation, brokering and service deployment architecture as proposed in Figure 1. In the following we explain the Meta-Negotiation infrastructure.

The registry is a searchable repository for meta-negotiation documents that are created by the participants. 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 existing client and/or service infrastructure, including, for example, negotiation or security clients. Besides being as a client for publishing and querying meta-negotiation documents (steps 1 and 2 in Figure 4), the middleware delivers necessary information for the existing negotiation clients, i.e. information for the establishment of the negotiation sessions (step 4, Figure 4) and information necessary to start a negotiation (step 5 in Figure 4).

4.2 Service brokering

In this subsection we are focusing on brokering-related aspects of the SRV architecture introduced in Section 2. Brokers (B) are the basic components that are responsible for finding the required services with the help of ASD. This task requires various activities, such as service discovery, matchmaking and interactions with information systems, service registries, repositories. There are several brokering solutions both in Grid [17] and SOAs [20], but agreement support is still an open issue. In our architecture brokers need to interact with ASDs and use adaptive mechanisms in order to fulfill the agreement (further requirements and interoperation is detailed in Section 4.3).

A higher-level component is also responsible for brokering in our architecture: the Meta-Broker (MB) [14]. *Meta-brokering* means a higher level resource management that utilizes existing resource or service brokers to access various resources. In a more generalized way, it acts as a mediator between users or higher level tools (e.g. negotiators or workflow managers) and environment-specific resource managers. The main tasks of this component are: to *gather* static and dynamic broker properties (availability, performance, provided and deployable services, resources, and dynamic QoS properties related to service execution), to *interact* with MN to create agreements for service calls, and to *schedule* these service calls to lower level brokers, i.e. match service descriptions (SD) to broker properties (which includes broker provided services). Finally the service call needs to be *forwarded* to the selected broker.

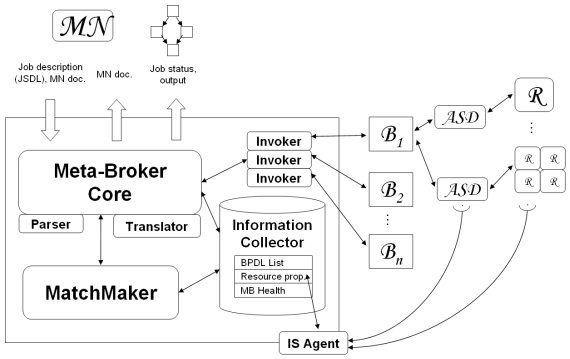

Figure 5: Meta-Broker in SRV.

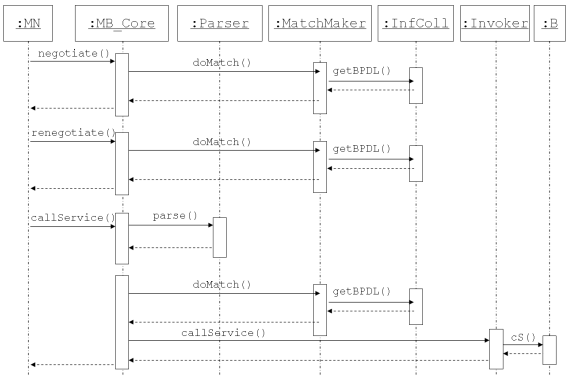

Figure 6: Interactions of the components of the Meta-Broker during utilization.

Figure 5 details the Meta-Broker (MB) architecture showing the required components to fulfill the above mentioned tasks. Different brokers use different service or resource specification descriptions for understanding the user request. These documents need to be written by the users to specify all kinds of service-related requirements. In case of resource utilization in Grids, OGF [1] has developed a resource specification language standard called JSDL [2]. As the JSDL is general enough to describe jobs and services of different grids and brokers, this is the default description format of MB. The *Translator* component of the Meta-Broker is responsible for translating the resource specification defined by the user to the language of the appropriate resource broker that MB selects to use for a given call. These brokers have various features for supporting different user needs, therefore an extendable Broker Property Description Language (BPDL) [15] is needed to express metadata about brokers and their offered services. The *Information Collector* (IC) component of MB stores the data of the reachable brokers and historical data of the previous submissions. This information shows whether the chosen broker is available, or how reliable its services are. During broker utilization the successful submissions and failures are tracked, and regarding these events a rank is modified for each special attribute in the BPDL of the appropriate broker (these attributes were listed above). In this way, the BPDL documents represent and store the dynamic states of the brokers. In order to support load balancing, there is an *IS Agent* (IS refers to Information System) reporting to IC, which regularly checks the load of the underlying resources of each connected broker, and store this data. It also communicates with the ASDs, and receives up-to-date data about the available services and predicted invocation times (that are used in the negotiations). The matchmaking process consists of the following steps: The *MatchMaker* (MM) compares the received descriptions to the BPDL of the registered brokers. This selection determines a group of brokers that can provide the required service. Otherwise the request is rejected. In the second phase the MM counts a rank for each of the remaining brokers. This rank is calculated from the broker properties that the IS Agent updates regularly, and from the service completion rate that is updated in the BPDL for each broker. When all the ranks are counted, the list of the brokers is ordered by these ranks. Finally the first broker of the priority list is selected, and the *Invoker* component forwards the call to the broker.

As previously mentioned, three main tasks need to be done by MB. The first, namely the information gathering, is done by the IS Agent, the second one is negotiation handling and the third one is service selection (illustrated in Figure 6). They need the following steps: During the negotiation process the MB interacts with MN: it receives a service request with the service description (in JSDL) and SLA terms (in MN document) and looks for a deployed service reachable by some broker that is able to fulfill the specified terms. If a service is found, the SLA will be accepted and the and MN notified, otherwise the SLA will be rejected. If the service requirements are matched and only the terms cannot be fulfilled, it could continue the negotiation by modifying the terms and wait for user approval or further modifications.

4.3 Service deployment and virtualization

Automatic service deployment (ASD) is a higher-level service management concept, which provides the dynamics to SBAs – e.g. during the SBA's lifecycle services can appear and disappear without the disruption of their overall behavior.

Figure 7 shows the ASD's related components to the SRV architecture and their connections. To interface with a broker the ASD should be built on a repository (as an example it can use the Application Content Service – ACS [9] – standard proposed by the OGF [1]). All the master copies of all the deployable services should be stored in the repository. In this context the master copy means everything what is needed in order to deploy a service on a selected site – which we call the virtual appliance (VA). The virtual appliance could be either defined by an external entity or the ASD solution should be capable of acquiring it from an already running system. The repository allows the broker to determine which services are available for deployment and which are the static ones. Thus the repository would help to define a schedule to execute a service request taking into consideration those sites where the service has been deployed and where it could be executed but has not yet been installed (it is also monitored by the IS Agent of the Meta-Broker detailed in Section 4.2). If the deployed services are not available, it checks whether any of the latter resources can deliver the service taking into account the deployment cost.

The *Workspace Service* (WS), offers the virtualization capabilities – virtual machine creation, removal and manage-

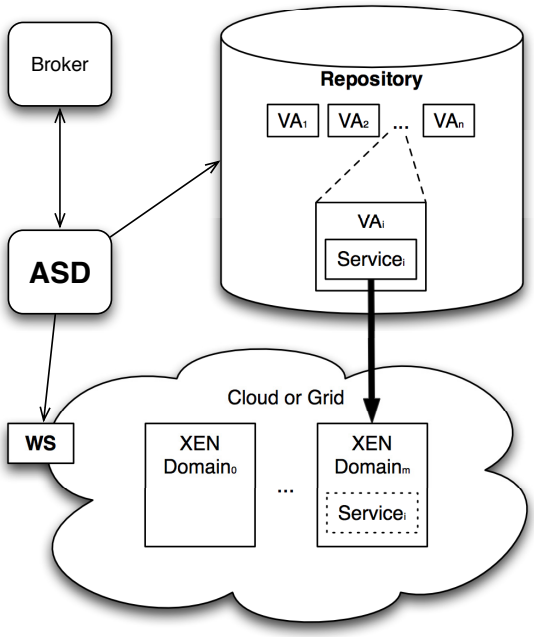

Figure 7: Service deployment in SRV.

ment – of a given site as a WSRF service. According to the OGSA-Execution Planning Services (EPS) [8] scenarios, a typical service broker has two main connections with the outside world: the *Candidate Set Generators* (CSG), and the Information Services. The task of the CSG is to offer a list of sites, which can perform the requested service according to the SLA and other requirements. Meanwhile the information services should offer general overview about the state of the SBA, Grid or Cloud. In most of the cases the candidate set generator is an integral part of the broker thus instead of the candidate set adjustments, the *broker* queries the candidate site list as it would do without ASD. Then the broker would evaluate the list and as a possible result to the evaluation it would initiate the deployment of a given service. As a result the service call will be executed as a composed service instead of a regular call. The composition will contain the deployment task as its starting point and the actual service call as its dependent task. Since both the CSG and the brokers heavily rely on the *information system*, the ASD can influence their decision through publishing dynamic data. This data could state service presence on sites where the service is not even deployed.

The selected *placement* of the ASD depends on the site policies on which the brokering takes place. In case the site policy requires a strict scheduling solution then either the CSG or the information system can be our target. If there is no restriction then the current broker can be replaced with an ASD extended one. In case the *candidate set generator* is altered then it should be a distributed, ontology-based adaptive classifier to define a set of resources on which the service call can be executed [28]. The CSG can build its classification rules using the specific attributes of the local information systems. Each CSG could have a feedback about the performance of the schedule made upon its candidates. The ASD extended CSG should have three interfaces to interoperate with other CSGs and the broker. First of all, the CSGs could form a P2P network, which requires two interfaces. The first manages the ontology of the different information systems by sharing the classifier rules and the common ontology patterns distributed as an OWL schema. The second interface supports decision-making among the peers. It enables the forwarding of the candidate request from the broker. The third interface lies between the broker and the CSGs to support passing the feedback for the supervised learning technique applied by the CSGs. This interface makes it possible for the broker to send back a metric packet about the success rate of the candidates.

The *brokers* should be adapted to ASD differently depending on where the ASD extensions are placed. If both the CSG's and the broker's behavior is changed then the broker can make smarter decisions. After receiving the candidate site set, the broker estimates the deployment and usage costs of the given service per candidate. For the estimation it queries the workspace service (WS). This service should accept cost estimate queries with a repository entry (VA) reference as an input. The ASD should support different agents discovering different aspects of the deployment. If only the broker's behavior is changed, and the CSG remains untouched, then the ASD would generate deployment service calls on overloaded situations (e.g. when SLA requirements are endangered). These deployment service calls should use the workspace service with the overloaded service's repository reference.

Finally it is possible to alter the *information system's behavior*. This way the ASD provides information sources of sites, which can accept service calls after deployment. The ASD estimates and publishes the performance related entries (like estimated response time) of the information system. These entries are estimated for each service and site pair, and only those are published which are over a predefined threshold.

Regarding component interactions, the ASD needs to be extended with the following in order to communicate with brokers: Requested service constraints have to be forced independently from what Virtual Machine Monitor (or hypervisor [3]) is used. To help the brokers making their decisions about which site should be used the ASD has to offer deployment cost metrics which can even be incorporated on higher level SLAs. The ASD might initiate service deployment/decommission on its own when it can prevent service usage peaks/lows, to do so it should be aware of the agreements made on higher levels.

5. CASE STUDY

In this section we discuss a case study on the *Maxillo Facial Surgery Simulation* (MFSS) in order to demonstrate the utilization of the presented architecture. 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 running on local and different remote machines. These components may be organized as a workflow in order to simplify the work of the end users [4]. The main steps of the simulation are: (i) mesh generation is used for the generation of meshes necessary for the finite element simulation; (ii) mesh manipulation defines the initial and boundary conditions for the simulation; (iii) finite element analysis is a fully parallel MPI application usually running on a remote HPC cluster. In the followings we describe step by step how MFSS can

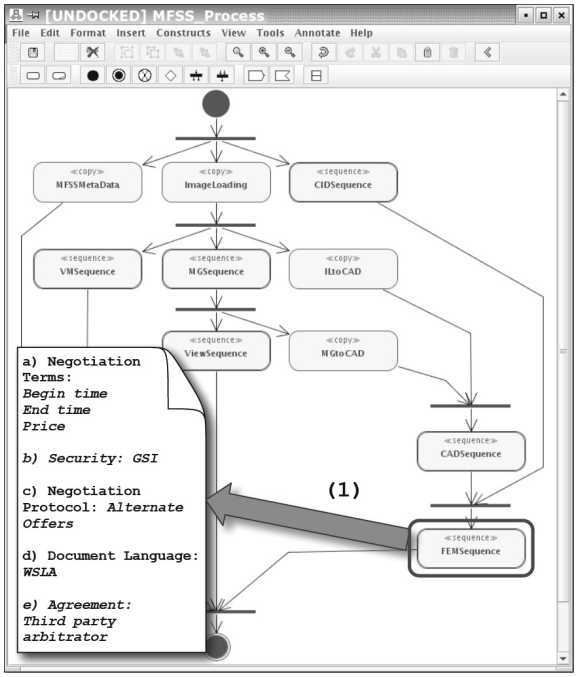

Figure 8: MFSS workflow with meta-negotiation specification.

be executed on the proposed architecture for the SLA-based Resource Virtualization. MFSS application can be modeled and executed using the Amadeus workflow tool, a QoS-aware Grid modeling, planning, and execution tool [4], see Figure 8.

As depicted in Figure 8, 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 [7] security is required. The negotiation is performed based on the alternate offers protocol [32]. 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 8, QoS is specified using WSLA [13]. The constraints shown in Figure 8 are transformed into a XML based meta-negotiation document. Thereafter this document is passed to the Meta-Broker. During the execution of the workflow, the Meta-Broker receives the service description in JSDL and the SLA terms in the meta-negotiation document. First a matchmaking process is started to select a broker that is able to execute the job with the specified requirements (resource requirements and agreement terms). The broker with the best performance values is selected, and the description and agreement is translated to the format understandable by the broker. Thereafter the broker is invoked with the transformed descriptions. The selected broker receives the descriptions and calls the ASD to deploy a service on a Cloud or a Grid, taking into account the cost requirements of the agreement, or chooses an already deployed, idle computing service. The job is executed and the results are returned to the workflow enactor. Finally the ASD decommissions the service.

6. CONCLUSIONS

In this paper a novel architecture for SLA-based resource virtualization with on-demand service deployment is introduced. The solution incorporates three enhancements: a meta-negotiation component for generic SLA management, a meta-brokering component for diverse broker management and an automatic service deployment for resource virtualization on the Cloud. We have stated the essential requirements for building the target architecture and demonstrated the utilization through a future case study. Our future work aims at finalizing the presented components and interfacing the architecture to commercial Clouds and production Grids.

7. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube), and by the Vienna Science and Technology Fund (WWTF) under agreement ICT08-018, FoSII – Foundations of Self-governing ICT Infrastructures.

8. REFERENCES

[1] Open grid forum website. http://www.ogf.org, 1999.

[2] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job submission description language (jsdl) specification, version 1.0. Technical report, 2005. http://www.gridforum.org/documents/GFD.56.pdf.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[4] I. Brandic, D. Music, S. Dustdar, S. Venugopal, and R. Buyya. Advanced qos methods for grid workflows based on meta-negotiations and sla-mappings. In *The 3rd Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, November 2008.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 2009.

[6] M. Q. Dang and J. Altmann. Resource allocation algorithm for light communication grid-based workflows within an sla context. *Int. J. Parallel Emerg. Distrib. Syst.*, 24(1):31–48, 2009.

[7] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, New York, NY, USA, 1998. ACM.

[8] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel,

F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Reich. The open grid services architecture, version 1.5. Technical report, 2006. http://www.ogf.org/documents/GFD.80.pdf.

[9] K. Fukui. Application contents service specification 1.0. Technical report, 2006. http://www.ogf.org/documents/GFD.73.pdf.

[10] R. Howard and L. Kerschberg. A knowledge-based framework for dynamic semantic web services brokering and management. In *DEXA '04: Proceedings of the Database and Expert Systems Applications, 15th International Workshop*, pages 174–178, Washington, DC, USA, 2004. IEEE Computer Society.

[11] A. Iosup, T. Tannenbaum, M. Farrellee, D. Epema, and M. Livny. Inter-operating grids through delegated matchmaking. *Sci. Program.*, 16(2-3):233–253, 2008.

[12] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Sci. Program.*, 13(4):265–275, 2005.

[13] A. Keller and H. Ludwig. The wsla framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, V11(1):57–81, March 2003.

[14] A. Kertesz and P. Kacsuk. Meta-broker for future generation grids: A new approach for a high-level interoperable resource management. In *Grid Middleware and Services Challenges and Solutions*, pages 53–63. Springer US, 2008.

[15] A. Kertesz, I. Rodero, and F. Guim. Data model for describing grid resource broker capabilities. In *Grid Middleware and Services Challenges and Solutions*, pages 39–52. Springer US, 2008.

[16] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[17] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, 2002.

[18] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, Washington, DC, USA, 2004. IEEE Computer Society.

[19] Z. Li and M. Parashar. An infrastructure for dynamic composition of grid services. In *GRID '06: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pages 315–316, Washington, DC, USA, 2006. IEEE Computer Society.

[20] Y. Liu, A. H. Ngu, and L. Z. Zeng. Qos computation and policing in dynamic web service selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004.

[21] 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)*, February 2005.

[22] M. Parkin, D. Kuo, J. Brooke, and A. MacCulloch. Challenges in eu grid contracts. In *Proceedings of the 4th eChallenges Conference*, pages 67–75, 2006.

[23] D. M. Quan and J. Altmann. Mapping a group of jobs in the error recovery of the grid-based workflow within sla context. *Advanced Information Networking and Applications, International Conference on*, 0:986–993, 2007.

[24] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: Accountable execution of untrusted programs. In *In Workshop on Hot Topics in Operating Systems*, pages 136–141, 1999.

[25] N. G. G. Report. Future for european grids: Grids and service oriented knowledge utilities – vision and research directions 2010 and beyond. Technical report, December 2006. ftp://ftp.cordis.lu/pub/ist/docs/grids/ngg3_eg_final.pdf.

[26] I. Rodero, F. Guim, J. Corbalan, L. Fong, Y. Liu, and S. Sadjadi. Looking for an evolution of grid scheduling: Meta-brokering. In *Grid Middleware and Services Challenges and Solutions*, pages 105–119. Springer US, 2008.

[27] M. Surridge, S. Taylor, D. De Roure, and E. Zaluska. Experiences with gria – industrial applications on a web services grid. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 98–105, Washington, DC, USA, 2005. IEEE Computer Society.

[28] M. Taylor, C. Matuszek, B. Klimt, and M. Witbrock. Autonomous classification of knowledge into an ontology. In *The 20th International FLAIRS Conference (FLAIRS)*, 2007.

[29] A. Tsalgatidou and T. Pilioura. An overview of standards and related technology in web services. *Distrib. Parallel Databases*, 12(2-3):135–162, 2002.

[30] T. Vazquez, E. Huedo, R. S. Montero, and I. M. Llorente. Evaluation of a utility computing model based on the federation of grid infrastructures. In *Euro-Par 2007 Parallel Processing*, pages 372–381. Springer Berlin / Heidelberg, 2007.

[31] 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, 2006.

[32] S. Venugopal, X. Chu, and R. Buyya. A negotiation mechanism for advance resource reservation using the alternate offers protocol. In *16th International Workshop on Quality of Service (IWQoS 2008)*, pages 40–49, June 2008.