A Framework for managing Quality of Service in Cloud Computing through Service Level Agreements

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen
im Promotionsprogramm Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von

Wolfgang Ziegler
aus Mannheim, Deutschland

Hennef
Oktober 2016

Betreuungsausschuss

Prof. Dr. Ramin Yahyapour,
Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen mbH (GWDG),
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Jens Grabowski,
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Dieter Kranzlmüller,
Leibniz Rechenzentrum (LRZ),
Institut für Informatik der Ludwig-Maximilians-Universität München

Mitglieder der Prüfungskommission

Referent: Prof. Dr. Ramin Yahyapour,
Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen mbH (GWDG),
Institut für Informatik, Georg-August-Universität Göttingen

Korreferent: Prof. Dr. Jens Grabowski,
Institut für Informatik, Georg-August-Universität Göttingen

Weitere Mitglieder der Prüfungskommission

Dr. Lena Wiese,
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Caroline Sporleder,
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Xiaoming Fu,
Institut für Informatik, Georg-August-Universität Göttingen

Prof. Dr. Carsten Damm,
Institut für Informatik, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 11. Januar 2017

Acknowledgements

Research work that led to this thesis started almost a decade ago. Naturally, many people were involved in joint research during this time period. A number of them have directly or indirectly contributed to the work presented in this thesis and deserve acknowledgment. First of all, among the current (and former) colleagues in my research group and the various European and German projects through which a significant part of the presented research was funded, I would like to thank Oliver Wäldrich, Philipp Wieder, Angela Rumpl for fruitful fundamental discussions, and (in no particular order) Hassan Rasheed, Karl Catewicz, Alexander Papaspyrou, Josep Martrat, Ana Ferrer, Vincent Keller, Uwe Schwiegelshohn, Michael Parkin, Antoine Pichot, Dominic Battré, Alexander Fölling, Johann Tordsson, and Raül Sirvent for all their contributions. At the Open Grid Forum I would like to thank the members of the GRAAP-WG for valuable discussions, in particular Toshiyuki Nakata, Bastian Koller, Frances Brazier, Kassidy Clark, Michael Oey, Karl Czajkowski, and Heiko Ludwig. Big gratitude towards my institute and department, in particular Martin Hofmann-Apitius, for their support of the writing phase of this thesis. Last, but not least, I would like to thank my advisor Ramin Yahyapour and my co-advisors Jens Grabowski and Dieter Kranzlmüller for inspiring discussions and support for finalising this dissertation.

Abstract

Until today, Cloud providers only offer a limited set of non-negotiable service levels to their customers. Most often these service levels are expressed as guarantees for availability together with the offer to have access to a virtualised environment with a certain performance the customer may select from a number of predefined configurations. This simplifies the life of the provider, e.g., in terms of effort to maintain an adequate infrastructure, or regarding the effort for reducing the risk violating Service Level Agreements (SLAs) with its customer. In consequence, the current practice is slanted towards the benefit of the provider and ignores more specific requirements of its customers, e.g. regarding data protection and related guarantees. An analysis of the underlying problems shows two major fields to be worked on for solving the problem: Firstly, each provider uses its own proprietary technology for managing SLAs throughout their life-cycles. However, first standards are available and could be employed allowing the customer to use a single standard interface to negotiate with several providers. Secondly, there is neither a common set of terms to describe Cloud customers requirements regarding the Cloud services requested, nor, the back side of the medal, there is a common set of terms to describe the Quality of Service (QoS) of the Cloud providers' offerings. The focus of the presented work is (i) on the standard technology for negotiating and creating SLAs and (ii) the common terms and metrics describing providers' offerings and customers' requirements. Without these terms mapping the customers' requirements to cloud providers' offerings is a tedious manual and error-prone process and resulting SLAs will remain rudimentary. Additionally, both providers and their customers would benefit from more sophisticated and negotiable Service Level Agreements using existing standards. These SLAs are both (i) binding and monitorable agreements between the customer and the provider covering the customers' requirements and (ii) the basis for a QoS-aware Cloud resource management on the side of the provider including provisioning of physical machines and optimised allocation of virtual machines. Besides more traditional QoS aspects, terms related to Cloud Federation, Data Protection or Security Level Agreements are covered. Customers may use standards compliant agreement templates with the providers' offerings to select suitable providers for starting the negotiation to the extent the provider allows in the template. During the negotiation the provider may take into account the actual degree of capacity

utilisation of its infrastructure and active SLAs, and may use the SLA resulting from a successful negotiation to further optimise its infrastructure through application and VM consolidation. The work presented in this thesis covers about a decade of research starting in the environment of Grid computing and ending with today's Cloud computing.

Preface

This thesis comprises an introduction and the following papers:

Paper I — Battré, Dominic; Brazier, Frances M. T.; Clark, Kassidy P.; Oey, Michael; Papaspyrou, Alexander; Wäldrich, Oliver; Wieder, Philipp; Ziegler, Wolfgang: A proposal for WS-agreement negotiation. In: *Association for Computing Machinery -ACM-; Institute of Electrical and Electronics Engineers -IEEE-: 11th IEEE/ACM International Conference on Grid Computing 2010. Proceedings.* 25-29 October 2010 in Brussels, Belgium New York, NY: IEEE, 2010, S.233-241

Paper II — Ziegler, Wolfgang: SLAs for energy-efficient data centres: The standards-based approach of the OPTIMIS project. In: *Huusko, J.: Energy Efficient Data Centres: First International Workshop, E2DC 2012.* Madrid, Spain, May 8, 2012, Revised Selected Papers. Berlin: Springer, 2012. (Lecture Notes in Computer Science 7396), pp. 37-46

Paper III — Barnitzke, Benno; Ziegler, Wolfgang; Vafiadis, George; Nair, Srijith; Kousiouris, George; Corrales, Marcelo; Wäldrich, Oliver; Forgó, Nikolaus; Varvarigou, Theodora: Legal restraints and security requirements on personal data and their technical implementation in clouds. In: *Cunningham, P.: eChallenges e-2011. Proceedings. CD-ROM: Conference & Exhibition.* 26-28 October 2011 Florence, Italy. Dublin: IIMC, 2011

Paper IV — Cacciari, Claudio; Mallmann, Daniel; D'Andria, Francesco; Hagemeier, Björn; Rumpl, Angela; Ziegler, Wolfgang; Zsigri, Csilla; Martrat, Josep: SLA-based management of software licenses as web service resources in distributed computing infrastructures. In: *Future generation computer systems : FGCS.* Vol.28 (2012), No.8, pp.1340-1349

Paper V — Blasi, Lorenzo; Jensen, Jens; Ziegler, Wolfgang: Expressing quality of service and protection using federation-level service level agreement. In: *Euro-Par 2013. Parallel Processing Workshops: BigDataCloud, DIHC, FedICI, HeteroPar, HiBB, LSDVE, MHPC, OMHI, PADABS,*

PROPER, Resilience, ROME, and UCHPC 2013. Aachen, Germany, August 26-27, 2013; revised selected papers. Berlin: Springer, 2014. (Lecture Notes in Computer Science 8374), pp. 146-156

Paper VI Juan Ferrer, Ana; Hernández, Francisco; Tordsson, Johan; Elmroth, Erik; Ali-Eldin, Ahmed; Zsigri, Csilla; Sirvent, Raül; Guitart, Jordi; Badia, Rosa Maria; Djemame, Karim; Ziegler, Wolfgang; Dimitrakos, Theo; Nair, Srijith K.; Kousiouris, George; Konstanteli, Kleopatra; Varvarigou, Theodora; Hudzia, Benoit; Kipp, Alexander; Wesner, Stefan; Corrales, Marcelo; Forgó, Nikolaus; Sharif, Tabassum; Sheridan, Craig: OPTIMIS: A holistic approach to cloud service provisioning. In: *Future Generation Computer Systems : FGCS.* Vol.28 (2012), No.1, pp.66-77

Paper VII Birkenheuer, Georg; Brinkmann, André; Högqvist, Mikael; Papaspyrou, Alexander; Schott, Bernhard; Sommerfeld, Dietmar; Ziegler, Wolfgang: Infrastructure federation through virtualised delegation of resources and services In: *Journal of Grid Computing.* Vol.9 (2011), No.3, pp.355-377

The research presented in this thesis was conducted using resources of the Department of Bioinformatics at the Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI). Several German and European projects provided financial support: the European Commission (EC)'s Sixth and Seventh Framework Programme under grant agreements: 004265 (FP6, CoreGrid) 257115 (FP7, OPTIMIS), 216759 (FP7, SmartLM), and the German Federal Ministry of Education and Research (BMBF) under project grants 01IG09009 (DGSI), 01AK800A (SLA4D-Grid).

Contents

Abstract **i**

Preface **iii**

1 Introduction **1**
1.1 Motivation 1
1.2 Research questions, aims and objectives of the dissertation 3
1.3 Approach to address the research questions 4
1.4 Developments 5
1.5 Structure of the dissertation 7

2 Grids, Clouds, Quality of Service and Service Level Agreements – Concepts, Definitions and Terminology **9**
2.1 Grids and Clouds 9
2.2 Service Level Agreements, Quality of Service 10
2.2.1 Evolving technology 10
2.2.2 Service Level Objectives, Key Performance Indicators, and monitoring 12
2.2.3 Current commercial practice 13
2.2.3.1 Cloud user's requirements regarding Service Level Agreements 14
2.2.4 Legal issues 15

3 Dynamic Machine-Processable Service Level Agreements **17**
3.1 Evolution of dynamic machine-processable SLAs 18
3.2 Research and development focussed on dynamic SLAs 21
3.3 More examples of application of dynamic SLAs 23

4 Term framework for Service Level Agreements **27**
4.1 Early implementations of Service Description Terms 28
4.2 OPTIMIS Service Manifest 28
4.3 Current work on Cloud Service Description Terms 31

5 Electronic Negotiation of Service Level Agreements **33**
5.1 Negotiation models 33
5.1.1 Agent-based negotiations 34
5.1.2 WS-Agreement-based negotiation 35
5.1.3 Other negotiation approaches 37
5.2 Preliminary work on a standard negotiation protocol 38
5.3 Progress towards WS-Agreement Negotiation 40

6 Service Level Agreements in Distributed Computing 43
6.1 Service Level Agreements in Grid computing . . . 43
6.1.1 Co-allocation of resources . . . 44
6.1.2 Activity or resource delegation . . . 45
6.2 Service Level Agreements in Cloud computing . . . 48
6.3 Grid and Cloud Broker . . . 50

7 Summary of Contributions 51
7.1 Dynamic electronic Service Level Agreements . . . 51
7.2 Term framework for Service Level Agreements . . . 52
7.3 Electronic Negotiation of Service Level Agreements . . . 52
7.4 Service Level Agreements in distributed computing . . . 52
7.4.1 Service Level Agreements in Grid computing . . . 52
7.4.2 Service Level Agreements in Cloud computing . . . 53
7.5 Paper overview . . . 53
7.5.1 Paper I . . . 53
7.5.2 Paper II . . . 54
7.5.3 Paper III . . . 54
7.5.4 Paper IV . . . 55
7.5.5 Paper V . . . 55
7.5.6 Paper VI . . . 56
7.5.7 Paper VII . . . 56

8 Conclusions 59

References 61

Paper I 77

Paper II 89

Paper III 101

Paper IV 113

Paper V 125

Paper VI 137

Paper VII 151

A Annex 177
A.1 Acronyms . . . 178
A.2 Service Manifest Schema . . . 181
A.3 Curriculum Vitae . . . 195

1. Introduction

During the last years all aspects of SLAs have become targets of research and development projects, have become under scrutiny of research agencies, and have become subject of standardisation. This thesis presents research results regarding machine-processable SLAs and our contributions to these. In particular, it presents the technology required to define, manage, and match the QoS both from the side of the providers' offerings and their customers' requirements. This work aims at promoting base technology well suited to transcend the current situation where QoS is defined by the providers and most often limited to an availability metric, which reduces the options of their customers to "take it or leave it" [15]. The other side of the problem is that today's SLAs are not machine-processable, hence, inhibiting automated provider selection based on their offered SLAs and the automatic negotiation and creation of SLAs, e.g. by agents like brokers. This work is consolidating the outcome of developments for machine-processable SLAs their automated negotiation and creation. Contributions are made to different aspects of the problem such as defining a set of terms to describe QoS beyond availability, a language and a protocol to create SLAs , and a protocol for multi-round negotiation of SLAs.

1.1 Motivation

The number of products, or more generally resources, that are accessible through computer-based services has been constantly increasing over the last years. The notion of e-business. was coined to distinguish the emerging way of offering and accessing these products from traditional businesses. This evolution has changed the modern business world in such a way that e-business has either a direct or indirect impact on all enterprises, public administrations and other organisations and, naturally, also affects their customers, users, or members.

Customers are used to comparing multiple similar products offered by different vendors in a real market place and to differentiate them, e.g., through visual or haptic examination. This enables customers to select and purchase the products that meet their demand regarding the quality of the products. In contrast, computer-based services in general do not allow visual or haptic examination. This raises the question

how similar computer-based services can be compared and differentiated to finally select the one that provides the quality that meets the demand of a customer.

Over time, using an SLA to describe and define the quality of a service has succeeded as a common and widespread approach in certain businesses. In fact, SLAs have been around for a quite some time. In the 1960s, they were the general operating procedures for achieving defined service levels and responding to service problems to which a company or an organisation agreed when buying or renting machine time on a mainframe. The big iron was the enterprise system by default, as no other technologies could match its processing capabilities, and able to satisfy the computing demand provided that the SLA was set-up accordingly and fulfilled.

Moreover, SLAs have been used in the telecommunication sector for more than 30 years to establish contracts governing the QoS of networks and network access with their commercial customers and between the providers regarding, e.g., bandwidth, jitter, or packet loss. Similar, SLAs between Information Technology (IT) departments and their internal users have been used for defining the QoS of the IT services provided, e.g., transaction rates, amount of time to retrieve data from databases, reaction and resolution times in case of IT service degradation or outage. In the recent past SLAs have been broadly used do determine QoS for all kind of services, e.g., in enterprise production environments, hospitals, public administration. In the telecommunication sector the initially paper based SLAs started to be transformed into machine-processable SLAs over time due to the increased flexibility required by customers and the more complex and dynamic provisioning.

One of the environments where SLAs have been used was Grid computing that started broadening the way of IT resource provisioning in academic environments 15 years ago. Grid computing became a convenient and broadly used way to satisfy temporary resource demand beyond the local capacity by sharing resources, especially for application in High Performance Computing (HPC) and High Throughput Computing (HTC). After several years of experience with voluntary contributed Grid resources being offered on a best effort basis with no particular QoS assurances, SLAs were considered beneficial for resource selection based on matching user requirements and resource capabilities, resource co-allocation from multiple providers and for guaranteeing the required QoS. However, there were no **common terms for describing the properties** of the resources offered by the participating organisations and the QoS required by the customer that could be used to identify the Grid resource with the most suitable properties, and to negotiate and create an SLA with guarantees for the required QoS. Additionally, **interoperable tools for negotiating and creating SLAs** didn't exist.

With the advent of Cloud computing 10 years ago, sharing of resources in academia was complemented by commercial services. The Cloud service providers started offering **non-negotiable** SLAs to their customers describing the QoS of their services. The SLAs focussed on a provider specific definition of an availability metric and included rudimentary compensations in case the availability was below defined thresholds. With Cloud computing being used more broadly the SLAs turned out to be too limited for many users while still being convenient for the providers, e.g., because of the simple management.

The emerging Cloud services offered by a multitude of providers with each enforcing their **proprietary SLA** are especially targeting Small and Medium Enterprises (SME) and end users. It turned out that matching the requirements of Cloud service customers, e.g. beyond a standard environment for hosting a less frequented web server, with the QoS offered by the Cloud service providers is cumbersome to impossible given the SLAs offered by the providers until today. For example, Cloud-based multi-user computer games have properties of an interactive real-time application and require strong QoS guarantees which clearly transcend the availability guarantees offered by Cloud providers [55]. However, similar to the situation in the Grid there were no **common terms for describing the QoS** offered by the provider and the QoS required by the customer that could be used to identify the provider with the most suitable service offering and to negotiate and create an SLA with guarantees for the required QoS. And **interoperable tools for negotiating and creating SLAs** didn't exist either.

Since their introduction, SLAs in Grid computing always have been machine-processable. In contrast, SLAs of Cloud providers are published on their web pages as unstructured text (and sometimes complemented by a written framework contract). For other application domains SLAs still are paper based.

1.2 Research questions, aims and objectives of the dissertation

Grid and Cloud computing can be considered advanced embodiments of distributed computing. While they are providing elementary access to resources at different levels of abstraction governed by different and more or less rich models for, e.g., access control, collaboration, security, or payment, it seems in general difficult to describe the required QoS and to get access to services or resources with a defined and guaranteed QoS.

In view of requirements and shortcomings as described in previous Section 1.1 we have identified three research questions whose answers contribute to closing the gaps. These are the starting points for the research described in this thesis:

1. What kind of technology is needed for creating machine-processable SLAs for arbitrary environments?
   The answer to question 1 requires a domain-agnostic solution providing a language for creating SLAs with a simple request-response protocol approach. The solution should be domain-agnostic for reasons of having an all-purpose language that can be used in different environments instead of an unmanageable bunch of domain-specific languages. Additionally, the language should be standardised for reasons of interoperability which would allow customers creating SLAs with different providers without being forced to use provider-specific technology.
   A language with the described propertied and the integrated protocol did not exist prior to the preliminary work for this thesis.

2. What language is needed for describing the quality of offered services and user requirements in an interoperable way?

In contrast to 1, addressing question 2 will require the development of domain-specific languages that capture the specifics of an environment with respect to QoS of offered services and user requirements. The language should easily integrate with outcome of 1, i.e. preferably it should be defined using the same base technology. Similar to question 1 these languages or more precisely the terms should undergo standardisation, e.g. to allow comparing the offerings of different providers and to allow creating an SLA with the most suitable one for a given task or business.
Such languages did not exist prior to the preliminary work for this thesis.

3. What are the required properties of a protocol to negotiate SLAs based on the outcome of 1 and 2?
While using the outcome of 1 would be convenient for many situations where SLAs can be created in a single step there are situations which require several rounds of negotiation before an agreement can be reached.
A suitable protocol for multi-round negotiation did not yet exist prior to the developments presented in this thesis.

In short, this thesis aims at presenting technology developed as solution for the three research questions: software for negotiation and creation of machine-processable SLAs for dynamic resource provisioning with changing providers in distributed environments. The application of this technology enables a framework for managing QoS in Cloud computing through SLAs.

1.3 Approach to address the research questions

With the changing paradigm how IT infrastructure is provisioned, companies' IT infrastructure is moving from hardware managed on premises to Cloud-based IT infrastructure services that could be served by an internal provider (private Cloud) or from a provider outside the company (public Cloud or hosted private Cloud), or combinations of both. Since one advantage of turning IT infrastructure into a service is the increased flexibility regarding provider selection and resource usage, traditional paper SLAs to manage QoS are too static and cannot be used in a dynamic service environment in consequence. Instead, dynamically created machine-processable SLAs are needed for managing QoS.

It is considered mandatory that the developments of machine-processable SLAs seamlessly integrate into web service technology, given the almost ubiquitous use of web service technology to access computer-based services, e.g. Cloud or Grid services.

As depicted in Figure 1.3.1 the framework for managing QoS in Cloud computing through SLAs comprises three building blocks: ***SLA creation***, ***SLA terms***, and ***SLA negotiation***. Although these components have been developed independently over time, they build upon each other to deliver the essential building blocks for machine-processable SLAs.

SLA creation is the basic component making available the domain-agnostic language and a simple request-response protocol for creating SLAs. It provides the solution to research question 1 and is described in Chapter 3 in more detail. Working on this solution led to the specification of WS-Agreement. I chaired the

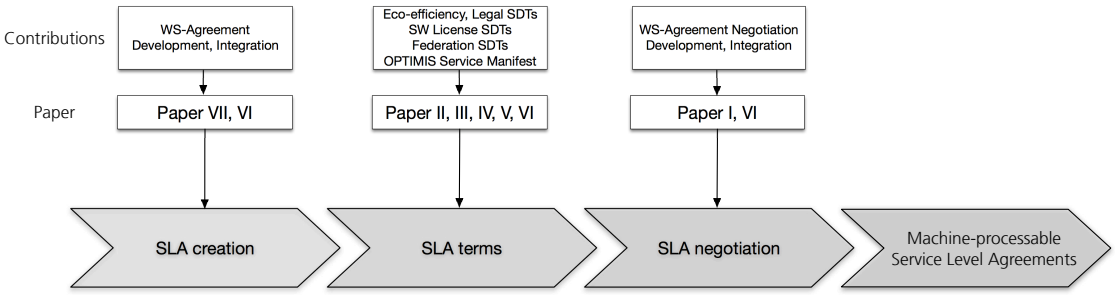


Figure 1.3.1: General view of the approach with related contributions and papers

development of WS-Agreement in the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG) of the Open Grid Forum (OGF) and contributed to the integration of WS-Agreement in various environments, like the D-Grid Scheduler Interoperability (DGSI) and the SLA4D-Grid developments of the German D-Grid, and the OPTIMIS CloudQoS component. The related papers VII and IV present integration and use of WS-Agreement in a Grid computing environment where activity and resource delegation is governed though SLAs, and in the software license management framework *elasticLM*®. Paper VI presents the OPTIMIS toolkit (developed for Infrastructure as a Service (IaaS) Cloud environments) with its WS-Agreement-based CloudQoS component.

The component ***SLA terms*** defines the Service Description Terms (SDTs) that allow using the ***SLA creation*** component to create SLAs in different environments. Chapter 4 describes the work in more detail. I chaired and contributed to the development of the SDTs for energy efficiency, software licenses, Cloud federation and legal information concerning data protection, namely Binding Corporate Rules (BCRs), Standard Contractual Clauses (SCCs) and Intellectual Property Rights (IPRs). The related papers II, III, IV, V , and Paper VI show the evolution of the SDTs from preliminary term sets and their application specific domains towards the comprehensive OPTIMIS Service Manifest.

Finally, the ***SLA negotiation*** component extends the ***SLA creation*** component with the protocol to execute multi-round negotiations. Chapter 5 describes the work in more detail. I chaired and contributed to developing and writing the WS-Agreement Negotiation specification and to its integration into different environments, like the OPTIMIS toolkit. Paper I presents the protocol and its integration with WS-Agreement and an explanatory use-case for the application of SLA negotiation. Similar to the description of the ***SLA creation*** component, we refer to Paper VI which presents the OPTIMIS toolkit since WS-Agreement Negotiation was also part of the CloudQoS component..

1.4 Developments

In the course of the approach described in 1.3 the following software artefacts have been developed:

- For the ***SLA creation*** component a revised version of the OGF's WS-Agreement specification was produced with a number of error corrections in the non-normative part of the specification. Other related work includes the integration

into the D-Grid Scheduling Interoperability framework as described in detail in paper VII, and in *elasticLM*® as described in paper IV.

- For the ***SLA terms*** component a number of Extensible Markup Language (XML) schemas for SDTs in SLAs have been developed. Namely, the following schemas that didn't exist before

  - a schema for describing energy efficiency parameters of data centres and the corresponding certification, described in paper II. The schema supports both selecting Cloud providers according to the certified energy efficiency of their data centres and including energy efficiency into the SLA.

  - a schema to describe software licenses and their included features, described in paper IV. This schema is the basis for creating license tokens for the execution of license-protected software in Clouds.

  - a schema that allows defining prerequisites and properties of Cloud federation, described in paper V. This schema can be used for selecting the most appropriate providers to establish a federation and to stipulate essential properties of the federation in the SLA, e.g. regarding data protection.

  - a schema to support incorporating legal information like BCR, SCC and IPR into the SLA, described in paper III. BCR and SCC are crucial for assuring compliance with European data protection requirements while the definition of IPR is essential to secure the customer's rights regarding the outcome of processing using resources of a Cloud provider.

  All of the above schemas (except the schema to describe software licenses) have been consolidated in the OPTIMIS Service Manifest together with many other SDTs developed in the course of the OPTIMIS project to build a comprehensive schema for Cloud SLAs.

- For the ***SLAs negotiation*** component we developed WS-Agreement Negotiation, a novel protocol for multi-round negotiations which extends WS-Agreement in a compatible way without requiring changes in the specification. Basic essentials and details of the protocol are described in paper I. WS-Agreement Negotiation has been included into OGF's standardisation process. At the time of writing this thesis it is in the state of a proposed recommendation.

In conclusion, this thesis presents developments for dynamic machine-processable SLAs that allow managing QoS, e.g. of today's Cloud-based computing or storage services, in a domain-independent way. The approach chosen includes both the development of a domain-agnostic language and protocols for negotiating and creating SLAs and the development of specific terms for describing service offerings and service requirements applicable in distributed computing environments like Grids and Clouds. In parallel, the result of our research and development has been fed into the standardisation process of the OGF with the objective to get feedback from a greater community and to develop an interoperable solution with a stronger impact than an isolated development (see Chapter 7 for a discussion of the impact).

1.5 Structure of the dissertation

Contributions presented in this thesis focus on machine-processable SLAs and include own scientific publications addressing, e.g., development of a framework of SLA terms and metrics, a multi-round SLA negotiation protocol, usage of SLAs in Grid and Cloud environments, as well as software artefacts, e.g., the XML specification of the language and the protocol for creating SLAs, the specification of the protocol for multi-round negotiation of SLAs, the XML schema of the SLA term framework and an Application Programming Interface (API) to manage the terms.

The remainder of this thesis is structured as follows. Chapter 2 provides the background for the following chapters with a brief introduction of Grids and Clouds as the environment for this work and an introductory discussion of QoS and SLAs. The following three chapters present results of our work. Chapter 3 addresses the ***SLA creation*** component and presents prerequisites, approaches and the solution developed for dynamically creating SLAs leading to the WS-Agreement specification. Chapter 4 addresses the ***SLA terms*** component and gives an overview on structure and components of the term framework developed. Different aspects of the preliminary work are discussed in papers II and III. Chapter 5 focuses on the ***SLA negotiation*** component and describes approaches towards machine-processable negotiation of SLAs. The fundamental work is presented in paper I. Chapter 6 introduces how SLAs have been used in Grid computing and provides an overview on Cloud SLA research and development work. Cloud SLAs are studied in papers V and VI, paper VII presents our contributions to using SLAs in Grids for resource management in an academic environment (the German D-Grid), while paper IV studies the application of SLAs in the specific business segment software license protection of commercial applications. Chapter 7 summarises the contributions of this thesis, relates thesis contributions to the field, highlights the observed impact, and presents the included thesis papers. In Chapter 8 we conclude the thesis and discuss some possible directions for future work.

2. Grids, Clouds, Quality of Service and Service Level Agreements – Concepts, Definitions and Terminology

In this chapter we introduce Grids and Clouds being the environment for this thesis and provide an introductory discussion of QoS and SLAs.

2.1 Grids and Clouds

Satisfying the temporary peak resource demand of users by procuring enough infrastructure resources leaves the excess resources idling during the day-to-day business. In the middle of the 90s Internet was ubiquitous and research centres and universities were all connected through their national research networks. In this environment a first approach (called Metacomputing) to overcome the problems of expensive over-provisioning was making use of external resources from other research centers for peak loads. Examples are, e.g., in Germany the NRW-Metacomputing Initiative [20], and in the USA the Legion vision of a worldwide virtual computer [61] or the I-WAY high-performance distributed computing experiment [52]. The concept of Metacomputing was further refined and appeared worldwide as Grid Computing [53] in the late 90 under the paradigm of resource sharing.

The initial definition of Grids published by Foster and Kesselman defined: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [53] p 18. As Grid computing is based on resource sharing handling of authentication and authorisation is mandatory in order to control access to the shared resources. Foster and Kesselman reflected this in their extended definition of Grid computing: "The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering

strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs." [53] p 40. Grids comprise resources, e.g. computing resources or storage, voluntarily contributed by, e.g. research centers or universities, on the basis of mutual access to these resources. Access to these resources is controlled by Grid middleware systems like Globus [64] or UNICORE [154]. Sharing resources requires standards-based interfaces, APIs, data formats, protocols and more to work smoothly. The Open Grid Forum [103] appeared as voluntary organisation of the Grid community to develop the missing open standards.

The non-academic world was faced similar problems satisfying occasional peak demands. Commercial Application Service Provider (ASP) was an early way of outsourcing computational tasks using external hardware and software resources of an application service provider similar to the Software as a Service (SaaS) model in Clouds at a later point in time. The business model included both hardware and software usage in a pay-as-you-go manner.

After Amazon started in 2006 offering its Cloud service Elastic Compute Cloud (EC2) there is a plethora of commercial Cloud providers worldwide today. Most of them are offering their business to customers (predominantly SMEs) in the same country.

Cloud resource are offered most often as IaaS, Platform as a Service (PaaS) or SaaS with different levels of resource control for the customer and different level of support by the provider [7, 92]

While voluntary contributed Grid resources initially were offered on a best effort base with no particular QoS assurances in form of, e.g. SLAs, Cloud computing as a commercial offering comprises some minimalist SLAs (see Section 2.2.3).

2.2 Service Level Agreements, Quality of Service

A Service Level Agreement represents a documented agreement between a service user[1] and a service provider in the context of a particular service provision. Depending on the environment, more parties may be involved, e.g., a single user and multiple providers, or a single user and a broker and multiple providers.

2.2.1 Evolving technology

SLAs have been used by telecommunication providers for more than 30 years to establish contracts with their customers and between the providers. Between 2004 and 2005 the TeleManagement Forum (TMF) [150], the organisation of the telecommunication industries, has published three handbooks on SLA Management [1, 24, 136] that describe in detail concepts, principles, services and technology examples, and enterprise issues of end-to-end SLAs from enterprise to service provider. Naturally, their initial definition of an SLA has its roots in the ecosystem of telecommunication providers, however it already referred to concepts that can easily be generalised for other environments. The latest release of the SLA handbook was

[1] Since SLAs are not only used in commercial environments we use the more general term user instead of customer

published January 2011 [43]. The new definition of an SLA is no longer focussing on the telecommunication industry but provides a generic definition instead:

> "A Service Level Agreement (SLA) is an element of a formal, negotiated commercial contract between two parties, i.e. a Service Provider (SP) and a Customer. It documents the common understanding of all aspects of the service and the roles and responsibilities of both parties from service ordering to service termination. SLAs can include many aspects of a service, such as performance objectives, customer care procedures, billing arrangements, service provisioning requirements, etc. Although an SLA can address such items, the specification of the service level commitments on the SP part is the primary purpose of an SLA. Consequently, the concepts and principles in this handbook focus on the specification and management of SLAs, and on a framework for identifying quality and performance factors, i.e. for defining an appropriate Service Level Specification (SLS), including target numbers in the form of SLS Thresholds. ..." [43] p 13.

The first Cloud-focussed work on service metrics was published in 2013 by the National Institute of Standards and Technology (NIST). The document "proposes a possible framework to represent and use core metrology concepts as they relate to the measurement of Cloud services in order to establish sets of metrics, measures and units of measurement and their possible usage" [99] p 4, and is available from the NIST web site as Cloud Computing Reference Architecture Cloud Service Metrics Description. The work has been contributed to the standardisation process for Cloud SLAs that the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) started in 2013. The first part of a draft for an international standard was published in 2014 as ISO/IEC 19086-1: Cloud computing – Service Level Agreement (SLA) framework – Part 1: Overview and concepts [69]. Finally, the TMF initiated a collaboration of Standards Developing Organisations (SDOs) to identify contributions and gaps regarding end-to-end Cloud SLA management. Their report with a focus on use-cases, metrics and developments of the participating SDOs was published in 2014 [153]. It should be noted however, that neither of the three documents nor ISO/IEC19086-2 mentioned below addresses standardisation of terms that may be used in SLAs. Their focus is predominantly on models for metrics. **Standards for SDTs that would allow comparing provider offerings and selecting the most suitable one are still missing.**

Summing up:

- SLAs reflect an agreement between the provider and its user on the functional and non-functional QoS aspects of a particular service and define service levels of a service that must be maintained by a provider during service provision.
- Service properties are described using SDTs and are defined as a set of Service Level Objectives (SLOs). They need to be measurable and must be monitored during the provision of the service that has been agreed on in the SLA.
- Along with the SLOs, metrics have to be used that define conditions and rules for performing the measurement and for understanding the results of a measurement. These metrics usually are defined proprietarily (and hence are lacking interoperability) and agreed upon by the parties but in future may be

defined based on a standard such as the upcoming ISO/IEC standard[2] for a Metric Model [70].

- Besides descriptive SDTs and SLOs, guarantees are required in an SLA to achieve bindingness. Key Performance Indicators (KPIs) allow to measure whether the QoS promises of the SLA as defined in the guarantees are met or violated.

2.2.2 Service Level Objectives, Key Performance Indicators, and monitoring

The specific QoS attributes of a service must be agreed on between the user and provider(s), before service provision begins. Besides the obligations of the provider they may also define the obligations of the user when the service of the provider meets the quality specified in the SLA. The SLA should also contain a set of penalty clauses (or remedy clauses) in case the service provider fails to deliver the agreed quality (or exceeds the agreed quality). The penalty clauses are taken into account when the evaluation of the KPI indicates that an SLO has not been achieved. Depending on the nature of the SLO the evaluation of the KPI may be based on a single measurement or may take several measurements at different times into account to determine whether the SLO has been achieved. Similarly to penalties, rewards (remedies) for the provider can be defined in the SLA in case the provider overachieves the agreed service level(s). Both in case of dispute and for monitoring the service at runtime involvement of a mutually trusted third party may be necessary. Stamou et al. describe in more detail the concept of a Trusted Third Party (TTP) and requirements to build a TTP in the context of service transactions in Cloud environments [147]. Automating this conflict resolution process clearly provides substantial benefits over the current practice where the user both has to prove that the provider failed to meet the SLA and to claim the compensation defined in the penalty clauses (cf [34] p 5). Different outcomes from such an automated process are possible including monetary penalties, impact on potential future agreements between the parties and the enforced re-running of the agreed service. However, a number of concerns may arise when issuing such penalties, e.g., determining whether the service provider is the only party that should be penalised, or determining the type of penalty that must be applied to each party.

An SLA goes through various stages as depicted in Figure 2.2.1. Although sometimes more stages are attributed to the SLA lifecycle, e.g. identification and selection of the provider, agreeing on the terms of the SLA, destroying the SLA (cf. [89] p 262-265), we consider the depicted lifecycle as necessary and sufficient.

Figure 2.2.1: SLA lifecycle

[2] A draft version of the standard (Cloud Computing – Service Level Agreement (SLA) Framework — Part 2: Metrics, ISO/IEC19086-2) is expected to become available by end of 2016.

1. Provider-side development of a template describing the service offered by the provider.
2. The Customer starts negotiating with the provider about detailed terms of the SLA based on the template. This phase can either follow a discrete-offer protocol (a "take it or leave it" approach) or follows a multi-round negotiation protocol.
3. Deployment of the mutually agreed-upon-service
4. The Customer uses the service deployed in the previous phase according to the SLA. To verify that the service is delivered and used according to the SLA, monitoring is needed at this stage.
5. The achieved QoS is assessed and compared with the SLA after the agreed upon service period ends or once the provider terminates the service or the customer stops using the service. If defined in the SLA penalties or rewards will be handled here. Finally, the SLA is terminated at the end of this phase.

Obviously, SLAs sometimes need to be modified after having been accepted by the parties involved. For example, this may happen in case the resource situation of the provider changes, e.g., due to prioritised provisioning of resources to a premium customer, or because the resource demand of a customer changes. In the design phase the parties should decide whether an active SLA may be modified and if so include a process to change (re-negotiate) the Cloud SLA. The concept of mutable SLAs relates to the ongoing discussion about the question whether an SLA is a contract or not, which influences the way a modified SLA supersedes its predecessor. A further intricacy is owed to the fact that commonly accepted rules or directives for legally binding electronic agreements do not exist in Europe to date. Section 2.2.4 briefly recaps the positions.

2.2.3 Current commercial practice

The current state of SLAs offered by the providers is far from user requirements. As an example we present an SLA for Amazon EC2 (valid since June 2013) and contrast it with a fictitious but realistic user requirement. The SLAs of the other big providers are different in their wording but quite similar in their meaning, see the 2013 white paper "Public Cloud Service Agreements: What to Expect and What to Negotiate" of the Cloud Standards Customer Council for a comprehensive discussion [15]

Amazon EC2:

"If the Monthly Uptime Percentage is less than 99.95% but equal to or greater than 99.0%, that customer is eligible to receive a Service Credit equal to 10% of their bill (excluding one-time payments made for Reserved Instances) for the Eligible Credit Period. If the Monthly Uptime Percentage is less than 99.0% hat customer is eligible to receive a Service Credit equal to 30% of their bill (excluding one-time payments made for Reserved Instances) for the Eligible Credit Period.

We will apply any Service Credits only against future Amazon EC2 or Amazon EBS payments otherwise due from you." [48]

2.2.3.1 Cloud user's requirements regarding Service Level Agreements

We distinguish between a user with less interest or knowledge regarding technical details of the service and an experienced user that is in detail aware of the requirements of the service to be deployed into the Cloud.

The former will need an agreement template for a high-level SLA from the provider that allows selecting, e.g., a service class like Gold, Bronze, Silver, which will then be translated by the provider into a corresponding low-level SLA. While the latter will need an agreement template for a low-level SLA that allows selecting individual QoS properties of the service, e.g. performance or elasticity. However, both types of users may have common requirements to be included in the SLA that should be reflected in the template concerning, e.g., data security, data protection, BCR, SCC, or definition of the IPR regarding the results of the service (see [127] for details).

The EC2 example above shows, there has been little progress since the Open Cloud Manifesto group[3] [107] published in 2010 the Cloud Computing Use Cases White Paper [106]. The white paper identifies "two types of SLAs: Off-the-shelf agreements and negotiated agreements between a provider and a costumer to meet that consumer's specific needs." [106] p 54. Furthermore, it considers the off-the-shelf type as

> "unacceptable for any mission-critical application or data. ... Most public cloud services offer a non-negotiable SLA. With these providers, a consumer whose requirements aren't met has two remedies:
>
> 1. Accept a credit towards next month's bill (after paying *this* month's bill in full), or
> 2. Stop using the service.
>
> Clearly an SLA with these terms is unacceptable for any mission-critical applications or data. On the other hand, an SLA with these terms will be far less expensive than a cloud service provided under a negotiated SLA." [106] p 54-55.

The 2015 report of the CloudQuadrants initiative [36] concludes the state of practice: "The maturity of current used SLAs is not aligned with market demand" [35] p 3. Moreover, the report highlights:

> "The Standardisation Guidelines prescribe four themes of Service Level Objectives (SLO) to be included in state-of-the-art SLAs: Performance, Security, Data Management and Personal Data Protection. Per theme the guidelines expect certain objectives to be covered in an SLA.
>
> In general, the research shows that the currently used SLAs barely match these themes and their objectives. These SLAs have more in common with old standard outsourcing contracts that have been 'cloudified' to match the emergence of cloud technology. This is not by definition a shocking result, because these SLAs are still significantly better than the IT-outsourcing contracts for non-cloud services some organisations are still engaged in." [35] p 3.

The targeted automated management of SLAs has prerequisites including

[3] supported by companies like IBM or Rackspace (but for obvious reasons not by Cloud providers)

- the provider maintains a repository with machine-processable agreement templates from which the user can select the most appropriate one for making an offer to the provider for using a specific service
- the template includes SDTs[4] and probably creation constraints regarding the values the various terms may take in a concrete agreement.
- the provider offers an API for users that allow uploading a (probably modified) template to make an offer
- the provider is able to process the offer, i.e. accept or reject it or start a multi-round negotiation with the user to probably reach an agreement
- the template includes KPIs for automated detection of deviations from the agreement (using monitoring information)
- service monitoring - either available through an API of the provider, or a trusted third party, or set-up by the user

2.2.4 Legal issues

The debate regarding legal bindingness of machine-processed SLAs has been accompanying the developments of electronic SLAs for a couple of years now. Even today, there is no common resolution regarding the two positions. To some extent the resolution also is missing due to the fact that there still is no European regulation in place that clarifies role and prerequisites of electronically created contracts.

There are two different views on contracts and SLAs: the definition of the IT Infrastructure Library (ITIL) and the definition of the TMF. ITIL V3 makes a distinction between SLAs and contracts: An SLA is "An Agreement between an IT Service Provider and a Customer. The SLA describes the IT Service, documents Service Level Targets, and specifies the responsibilities of the IT Service Provider and the Customer. A single SLA may cover multiple IT Services or multiple Customers." [73] "A Contract is a legally binding agreement between two or more parties. Contracts are subject to specific legal interpretations." [73]. While TMF defines "A Service Level Agreement (SLA) is an element of a formal, negotiated commercial contract between two parties, i.e. a Service Provider (SP) and a Customer. It documents the common understanding of all aspects of the service and the roles and responsibilities of both parties from service ordering to service termination. SLAs can include many aspects of a service, such as performance objectives, customer care procedures, billing arrangements, service provisioning requirements, etc. Although an SLA can address such items, the specification of the service level commitments on the SP part is the primary purpose of an SLA" [43], p 13. In contrast to the report on the results of a workshop on Cloud Computing Service Level Agreements organised by the European Commission [79] that tends to ITIL's separation of agreement and contract we support the TMF approach to identify also contractual properties in an SLAs. The rationale being both the fact that Cloud environments allow for highly dynamic provisioning of different services by customers and the ongoing efforts of the European Commission to update earlier directives for electronic contracts. NIST defines this self-service aspect of Clouds as "On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network

[4] using standardised terms

storage, as needed automatically without requiring human interaction with each service's provider." [8] p 2-1.

As a compromise providers could offer a basic (written) framework contract in writing to their customers that is used as a legal basis for the electronic SLAs for concrete services. With this approach dynamic electronic SLAs could inherit the legally binding character from the framework contract.

According to NIST a provider's service agreement has three basic parts: "(1) a collection of promises made to consumers, (2) a collection of promises explicitly not made to consumers, i.e. limitations, and (3) a set of obligations that consumers must accept." [8] p 3-1. The document identifies "four key promises to consumers: Availability [...], Remedies for Failure to Perform [...], Data Preservation [...], Legal Care of Consumer Information." [8] p 3-1. In customer initiated SLAs these obligations can appear as requirements a provider has to fulfil or to negotiate with its customer. However, as analysed in the following Chapter 3 and summarised in the NIST document: "Although the self-service aspect of Clouds as defined in the Section 2 implies that a consumer either: (1) accepts a provider's pricing and other terms, or (2) finds a provider with more acceptable terms, potential consumers anticipating heavy use of Cloud resources may be able to negotiate more favourable terms. For the typical consumer, however, a Cloud's pricing policy and service agreement are non-negotiable." [8] p 3-1.

3. Dynamic Machine-Processable Service Level Agreements

In this chapter we focus on the ***SLA creation*** component presenting IBM's Web Service Level Agreement (WSLA) and the WS-Agreement specification we developed in OGF's GRAAP-WG in Section 3.1, and research regarding dynamic SLAs in Section 3.2. The chapter is complemented with a discussion on the application of dynamic SLAs.

Grid and even more Cloud resources are available to users in large variety of different offerings. On the one hand, each provider's offering includes various resources with diverse properties, e.g., cpu clock frequency and memory of the nodes, inter-node network connectivity, Wide Area Network (WAN) connectivity, eco-efficiency, geographical location, cost, and more. Users, on the other hand, have similar diverse requirements regarding the resources they want to use. Up to now, matching user requirements and resource properties is usually done manually by the user, i.e. comparing different providers' offerings and selecting the most appropriate one for the own resource demand.

With the changing paradigm concerning how IT infrastructure is provisioned, companies' IT infrastructure is moving from hardware managed on premises to Cloud-based IT infrastructure services that could be served by an internal provider (private Cloud) or from a provider outside the company (public Cloud or hosted private Cloud), or combinations of both. Turning IT infrastructure into a service is beneficial because of the increased flexibility regarding provider selection and resource usage. Traditional paper-based SLAs to manage QoS are too static and cannot be used in a dynamic services environment in consequence. Instead, dynamically created machine-processable SLAs are needed for managing QoS.

Still, today's providers' SLAs offered in proprietary provider-specific formats are not machine-processable, hence, inhibiting automated provider selection through the offered SLAs and the automatic negotiation and creation of SLAs, e.g. by user applications or by agents like brokers. Cloud service providers are offering non-negotiable SLAs to their customers describing the QoS of their services. The SLAs

focus on a provider specific definition of an availability metric and included rudimentary compensations in case the availability was below defined thresholds. With Cloud computing being used more broadly the SLAs turned out to be too limited for many users, e.g. because the SLAs do not provide guarantees for QoS beyond availability. However, SLAs are still convenient for the providers, e.g. because of the simple management. Over the last years all aspects of SLAs have become target of research and development projects, have become under scrutiny of research organisations and agencies and have become subject to standardisation.

In the evolving environment with a constantly increasing number of (micro) services, where monolithic services are disappearing in favour of services that are compositions of small and agile services from different providers, manually managing QoS of the individual services is no longer feasible. Instead, QoS details of services should be hidden from the user in favour of high-level QoS parameters that are mapped to technical parameters in the technical space of the provider as Masche et al. argue identifying an increasing role of SLAs in B2B [91].

Monitoring of SLOs and related guarantees of an SLA are vital for a meaningful evaluation of the state of the SLA. Monitoring in Clouds generates a number of questions regarding, e.g., which party is monitoring, is the monitoring information detailed enough, do both parties trust the monitoring information. Maarouf et al. [89] propose monitoring of SLAs by a TTP which could avoid trust issues arising if either of the contractual parties provided the monitoring information. However, TTPs are still rare and more time is needed to make them more popular.

3.1 Evolution of dynamic machine-processable SLAs

With the growing importance of web service technologies as a means to provide ubiquitous network-based access to services, more commercial services became available. However, in contrast to the free of charge services with a best-effort approach regarding QoS as hitherto users of commercial services requested a guaranteed QoS. Around the year 2000 a group at IBM started working on the specification of WSLA [16, 77, 87] which is based on XML and defined as an XML schema [164]. WSLA allows the creation of machine-readable SLAs for services implemented using web services technology that defines service interfaces in the Web Services Description Language (WSDL). WSLA allows to define assertions of a service provider to perform a service according to agreed guarantees for IT-level and business process-level service parameters such as response time and throughput, and measures to be taken in case of deviation and failure to meet the asserted service guarantees, for example a notification of the service customer. The language is extensible and allows adapting to the specifics of a particular domain. Both the provider of a service and the user of this service can use WSLA.

The assertions of the service provider are based on a detailed definition of the service parameters including how basic metrics are to be measured in systems and how they are aggregated into composite metrics. In addition, a WSLA expresses which party monitors the service, third parties that contribute to the measurement of metrics, supervision of guarantees or even the management of deviations of service guarantees. Interactions among the parties supervising the WSLA are also defined.

The WSLA specification includes a set of standard extensions allowing to define complete agreements that relate to Web services and include guarantees for response time, throughput and other common metrics.

WSLA was a free offering of IBM that was used to some extent including proposals for extensions like support for collaboration agreements [97], integration with the Business Process Execution Language for Web Services [54] (BPEL4WS) [23], or usage in Cloud computing [116]. To some extent WSLA can be considered as preparative work that contributed concepts to the development of WS-Agreement. However, WSLA was not generic enough with respect to the objectives of the WS-Agreement development. Meanwhile, WSLA is no longer available from IBM's official web pages.

Similar to the way the consideration of QoS aspects of web services evolved, Grid computing resources had been made available from the cooperating organisations of regional, national or international Grids of computing resources on a best-effort basis initially. However, missing guarantees regarding QoS of computing resources turned out to be an issue in Grid computing, especially when different resources need to be available at a certain time for orchestrating or co-allocating multiple resources [10, 49, 160].

To meet this growing demand we initiated in 2002 the Grid Resource Allocation Agreement working group (GRAAP-WG) [58] in the Open Grid Forum [103], targeting on the development of a standard for SLAs. The accomplishment of this group was WS-Agreement [6]. The objective of the group was to define a standard language and protocol to create SLAs that could be used in arbitrary environments. Besides members from academia the working group included a number of members from companies like HP, Platform Computing, NEC or IBM of which the latter already had been involved in the specification of WSLA. The specification was published as a proposed recommendation of the OGF in 2007 [5], the WS-Agreement experience document [14] presenting code-independent interoperable implementation was published in 2010 and in 2011 the WS-Agreement specification became a full recommendation of the OGF.

The XML-based language of WS-Agreement and the protocol for advertising the capabilities of service providers through agreement templates can both be used for creating agreements between service consumers and providers, and monitoring agreement compliance. The XML schema of WS-Agreement defines the overall structure of an agreement document. In addition, the WS-Agreement specification defines a single-round protocol for negotiating and establishing agreements dynamically based on web services (a set of WSDL definitions). The negotiation comprises a binding SLA offer of the agreement initiator[1] (based on a template of the agreement responder) to the agreement responder which the latter can either accept or reject. In case the responder accepts the offer, both parties are bound to the obligations defined in the SLA, otherwise, in case of rejection, the agreement initiator is no longer bound to its SLA offer. It should be noted that WS-Agreement is fully symmetric with respect to the initiating party which could be the service consumer as above or the service provider.

[1]The offer is binding to allow providers to reserve the necessary resources to satisfy the offered SLA request.

The structure of a WS-Agreement XML schema is highly extensible. Agreement templates embody all the customisable aspects of an agreement to include domain-specific elements and properties, which is a major difference regarding WSLA. Another difference between WS-Agreement and WSLA is that in WS-Agreement metrics are defined in any structure required by a domain-specific extension.

There are several interoperable implementations of WS-Agreement in different languages as Java or C , including the Grid middleware Globus [12] and Unicore [129] and the Cloud broker CompatibleOne [3]. The Java-based reference implementation WSAG4J [163] has been used in many projects. Besides SOAP-based implementations there is a growing number of RESTful [128] implementations following the Representational State Transfer (REST) paradigm, e.g "A RESTful Approach to Service Level Agreements for Cloud Environments" [22], or the Open Cloud Computing Interface (OCCI) agreement [102].

The presentation of the two SLA approaches WSLA and WS-Agreement is complemented by two additional developments supporting machine-processable SLAs: the SLA description language SLAng [80] initially developed within the European TAPAS project [149], and CC-Pi [27] offering a theoretical frame-work for mapping SLAs to service constraints.

The focus of SLAng is to provide a language, which addresses the specification of "contractual" relationships between customers and Application Service Providers and by that allows for a clear definition of obligations on all involved partners with respect to the provided QoS. SLAng thereby distinguishes between vertical (application, hosting, persistence and communication) and horizontal (service, container and networking) Service Level Agreements, depending on the respective relationship nature. The specification [139] is provided on base of the Essential Meta Object Facility (EMOF) meta-model of the Object Management Group (OMG)).

The CC-Pi model is more tightly-coupled to the mechanics of negotiation, and does not address common constructs such as agreement party details or service interfaces. The work focuses on a theoretical model for creating the best-suited SLA from the offering of different providers but is does not address the language of the SLA itself. For this reason, CC-Pi addresses a different problem than our research. However, CC-Pi could be used in conjunction with WS-Agreement for selecting the best-suited provider based on their offerings published in agreement templates.

Apart from the approaches mentioned above there exist other ones for defining and managing SLAs comprising for example languages to express SLAs, solutions to offer and discover services with dedicated service levels, and frameworks for dynamic SLA negotiation, creation and monitoring. Saravanan and Rajaram present a comprehensive - though sometimes a bit fuzzy - survey in their 2015 study of Cloud service level agreements [132]. The report of Parkin et al. provides a comparison of SLA use in six of the European commission's FP6 projects [114].

Except for WS-Agreement all other approaches were not continued, most often because they were project-specific developments and the project ended which provided the financial support for the development. Thus, they have not created significant uptake but nevertheless some came up with concepts worth mentioning like the SLAng language or the NextGrid SLA. Except for SLAng other approaches are not even accessible any more via the respective project web site. However, in

comparison to WS-Agreement, SLAng lacks standard-compliance, tool support, and wide distribution.

3.2 Research and development focussed on dynamic SLAs

Over the last decade a number of research projects addressed SLAs using WS-Agreement as base technology, like the SLA@SOI project [142]. Some of them focussed on terms and syntax to define SLAs, some tried to extend WS-Agreement, while others used it for different applications. A comprehensive overview can be found in the 2013 report "Cloud Computing Service Level Agreements" published by the European Commission[79]. In the next paragraphs we first describe three research activities without own contribution followed by own research activities in a project focussing on SLA-based resource selection (Intelligent Application-oriented Scheduling (IANOS)) and in a study on improvements of wall clock time for the execution of workflows through advance reservation of resources. The sections concludes with more recent research conducted in two doctoral theses.

SLA* is an abstract syntax for SLA that was published by Kearney et al. [76]. It was developed in the SLA@SOI project (2009 - 2011) with the objective to be independent of underlying technologies. It is neither coupled to particular notions of service, nor to particular modes of expression. It is extensible to diverse scenarios. One goal of the project was to develop technology supporting automated mapping of high-level end-user SLAs (e.g. requesting a dedicated service class like bronze or gold) to low-level technical SLAs (to be considered by the provider in the deployment process) through a domain-independent SLA framework. The SLA management framework of the SLA@SOI project supports the definition and monitoring of Service Level Objectives across the vertical layers of the service provisioning stack. SLAs are managed throughout the complete service lifecycle, spanning the entire services stack from business layer to infrastructure layer. The project uses the WS-Agreement and WS-Agreement Negotiation specifications for agreement negotiation and creation [82, 143, 144]. The development has been discontinued after the end of the project. However, schemas and other software artefacts are still available through the project's web pages.

Müller et al. [96] argue that though WS-Agreement is considered the most significant language to specify SLAs it misses support for specifying temporal aspects of any functional or non-functional property of a service and the agreement itself. The authors suggest using a temporal Domain-Specific Language (DSL) that has been defined as a schema by the authors for WS-Agreement "to incorporate validity periods into WS-Ag descriptions, such as qualifying conditions associated to SLOs, template creation constraints during agreement creation process, or preferences over service properties" [96]. The WS-Agreement specification requires the definition of the lifetime for an agreement and recommends using the *Qualifying Condition* elements of the *Guarantee Terms* for specifying temporal conditions. Since WS-Agreement is domain agnostic by design, domain specific properties, e.g., like the specification of Mean Time Between Failures (MTBF) in the examples in the paper, must be defined - as the authors did - in domain specific schemas that may then be included as SDTs in the agreement template. The work has been carried out in

two projects (FEDER and Web-Factories) and has not been continued after the end of the projects and as a consequence the schemas for temporal conditions and for preferences are no longer available online.

The work of Goiri et al. [57] aims at supporting CPU-based guarantees in Cloud SLAs. The work was done in the context of the European project OPTIMIS [109] and proposes a resource-level metric for specifying fine-grain guarantees on Central Processing Unit (CPU) performance. The metric allows resource providers to dynamically allocate their resources to the different deployed services depending on their demand. To determine the demand of a customer's service, CPU usage is incorporated in the metric definition. However, false SLA violations due to the customer's service not using all its allocated resources are identified and neglected. The SLA framework is using an XML-based [166] description that combines both WS-Agreement and WSLA schema [167, 168] specifications. This work results in SLAs that are to some extent similar to WS-Agreement SLAs but due to the inclusion of WSLA elements the SLAs are no longer compliant to the standard.

In the IANOS project (2006 - 2009) [125] we aimed at developing a framework for the automated selection of the most appropriate computational resources from multiple Grids for a high performance computing application. The work was based on an integration of the Swiss Intelligent Scheduling System (ISS) [39, 62] (a system for application-oriented scheduling) and the Vertically Integrated Optical Testbed for Large Applications (VIOLA) Meta Scheduling Service (MSS) [49]. The outcome of the project was the IANOS resource broker. The broker identifies the best-suited resources based on application characteristics and available monitoring information regarding this application. Using on this information the resource broker was also capable of estimating the expected runtime which in turn allowed, e.g., to select the most cost-efficient resources among the best-suited ones using a cost function, the most energy efficient resources, or select resources according to a tradeoff between different criteria defined by the user. WS-Agreement-based SLAs were used for allocation and (if needed) co-allocation of Grid resources while the agreement covered guarantees on, e.g., QoS, price, eco-efficiency. A Job Submission Description Language (JSDL) [75] schema was used as term language to describe the computational resources [63, 78].

In [161] we study the possibility to improve the execution of a distributed workflow with respect to its turn-around time using SLA-based advance reservation. We compare the best-effort approach with an approach where the start times of the jobs are optimised regarding the reduction of the make-span of the entire workflow. For the best effort approach the first job of the workflow is scheduled according to the actual load of the provider's resources, the following jobs are scheduled when the respective preceding job has finished. Using SLAs with independent resource providers for scheduling the individual jobs at the time when the execution of the respective preceding job is expected to be completed. To achieve this we use advance reservation of resources (based on the estimated run-times of the jobs) realised through SLAs with the providers [88]. In order to guarantee the resource availability we model time requirements and dependencies in the SLA using the guarantees of a WS-Agreement template defined for that purpose [90]. Depending on the amount of resources required for the workflow we achieved up to 50% gain in turn-around time where workflows requiring fewer resources benefitted less or not at all.

More recently, two doctoral theses and several related papers have addressed advanced applications of dynamic electronic SLAs. The work of Yaqub [172] focusses on generic methods for adaptive management of SLAs in Cloud computing exploring the role of SLAs regarding their potential to build and maintain trust where services are increasingly interdependent. The thesis originated in the context of the SLA@SOI project [142] and develops generic methods to automate SLA lifecycle management providing adaptiveness with localisation achieved through policy based controls. The work considers business models, services and the delivery technology being independent concepts that can be tied through SLAs. Given the SLA@SOI background the thesis also provides an advancement of the application of WS-Agreement-based SLAs in Cloud computing.

The thesis of Lu [84] presents research and developments regarding planning and optimisation during the lifecycle of Cloud computing SLAs. Similar to the thesis of Yaqub the research work that led to this thesis was conducted as part of the SLA@SOI project. The thesis analyses the current state-of-the-art in SLA management and identifies challenges such as SLA representation for Cloud services, business-related SLA optimisations, or service outsourcing and resource management and concludes that a methodology for the management of the different phases of SLAs during its lifespan is needed for facilitating Cloud SLA management. The work results in models and approaches for SLA lifecycle management enabling automatic service modelling, negotiation, provisioning and monitoring. Given the SLA@SOI background this thesis - like the thesis of Yaqub - also provides an advancement of the application of WS-Agreement-based SLAs in Cloud computing.

In "Fault-tolerant Service Level Agreement lifecycle management in clouds using actor system" [86] Lu et al. present an approach for automated SLAs, i.e. electronic formalised representation of SLAs and the management of their lifecycle, based on autonomous agents. The paper presents the application of actor systems for monitoring of large sets of SLA. The authors argue that this is a realistic approach for the automated management of the complete SLA lifecycle, including negotiation and provisioning, but focus on SLA monitoring as the driver of scalability requirements. The application of actor systems allows separating the agreement's fault-tolerance concerns and strategies into multiple autonomous layers that can be hierarchically combined into a parallelised, management structure. The proposed approach is independent from the technology used for expressing and creating SLAs and could be used together with WS-Agreement standard.

3.3 More examples of application of dynamic SLAs

A survey we carried out in 2008 shows the extent to which SLAs based on WS-Agreement have been used in Grid resource management [134]. In another article in 2009 Pontz et al. [121] present the results of an evaluation of "Service Level Agreement Approaches for Portfolio Management in the Financial Industry". Wäldrich [159] provides a detailed discussion of approaches for orchestration of resources in distributed heterogenous Grids using dynamic SLAs. Below we first present two research activities without own contribution followed by own research activities for implementing SLA-based resource orchestration [10, 88], developments

in the SLA4D-Grid project [13] of the German D-Grid [42], and SLA-based management of software licenses [31]. In these projects SLAs are implemented using the WS-Agreement specification.

In "Semantic WS-Agreement-based partner selection" [105] the authors argue that "The current WS-Agreement specification is based on XML-based domain vocabularies and therefore limits the ability of matching the agreements to syntactical matching". They propose employing domain knowledge generated using ontologies and rules to extend the matching capabilities beyond simple string matching. For their approach they added the expression, predicate, parameter, and value tags, as defined in the WSLA specification. Further, they add OntConcept annotation tags to the original schema of the *Terms section* of the WS-Agreement specification. The addition of the OntConcept tag links these expression parameters directly to the concrete ontology concept. The developments include the implementation of a tool for matching providers and consumers based on WS-Agreement templates utilising Semantic Web technologies. While the proposed approach of using semantic matching for provider selection can result in more accurate matches regarding user requirements and provider capabilities the major weakness of the implementation is that it results in isolated applications due to the incompatible extensions of the schema for agreement templates. The work was carried out in the context of the METEOR-S project [94] and has been discontinued 2005 after the end of the project.

In their article "Enabling open Cloud Markets through WS-Agreement extensions" [130] the authors analyse requirements for tradable resource goods. Based on the results, they "suggest a detailed goods definition, which is easy to understand, can be used with many market mechanisms, and addresses the needs of a Cloud resource market. The goods definition captures the complete system resource, including hardware specifications, software specifications, the terms of use, and a pricing function" [130]. The authors propose a Computing Resource Definition Language (CRDL) that can be used as domain-specific extension of WS-Agreement. The aim of the extension is expressing the complexity of system resources in a single descriptor. At that time the authors expected that the resource descriptions with CRDL will be accepted by a large number of users. However, there has not been any significant uptake since as of today providers use proprietary resource descriptions.

In [88] we present two examples of using an early version of WS-Agreement for co-allocation of resources from multiple providers for a single service request in (i) the German project VIOLA [157] and (ii) in the Japanese Business Grid project. Requirements of the two examples regarding co-allocation are analysed and the respective experience implementing WS-Agreement for co-allocation of resources is described. Both examples consider advance reservation of resources needed for co-allocation as part of the QoS requirements of a service. The objective of the presented research was to investigate the benefit of using SLAs for advance reservation in two different implementations of Meta-Schedulers: the Global Grid Job Manager (GGJM) and the MSS [160]. with two different implementations of WS-Agreement. Another goal of the work was to identify requirements for the negotiation of SLAs and to feed them back to the GRAAP-WG to steer the development of a WS-Agreement negotiation extension.

In the article "Co-Allocating Compute and Network resources - Bandwidth on Demand in the VIOLA Testbed" [10] we argue that the execution of advanced ap-

plications can be improved when they run on appropriate resources selected from the different, heterogeneous resources available in Grids. Examples of such applications are distributed multi-physics simulations where multiple resources are needed at the same time, or complex work-flows where the resources are needed with some temporal dependencies [10, 122]. The goal of the work presented (which was done in the context of the VIOLA project) was to develop a framework for co-allocation of computing and network resources for different types of geographically dispersed distributed applications using SLAs for advance reservation of the resources. As applications and data are distributed across different computing centres QoS of the network connections between the resources is contributing to the overall performance[2] and needs to be taken into account. Hence, besides advance reservation of computing resources to support efficient execution of the applications, mechanisms that guarantee the availability of the network must be available. As with computing resources we consider reservation as promising approach for availability and performance guarantee. Network reservation is realised with the Allocation and Reservation of Grid-enabled Optical Networks (ARGON) [120] network resource management system. For the experiments we used the VIOLA testbed with UNICORE [148] as Grid middleware providing solutions for the orchestration of resources of different sites belonging to different administrative domains. The negotiation of agreements on resource usage (including advance reservation) with the individual local resource management systems is done by the MSS. WS-Agreement is used for creating the SLAs with the Local Resource Management System (LRMS) on the advance reservation of the resources. Once the SLAs are created the LRMS include the advance reservation in their individual schedules. ARGON acts as a LRMS for network resources and allows user or application driven selection and reservation of network connections with dedicated QoS based on network technologies like bandwidth on demand.

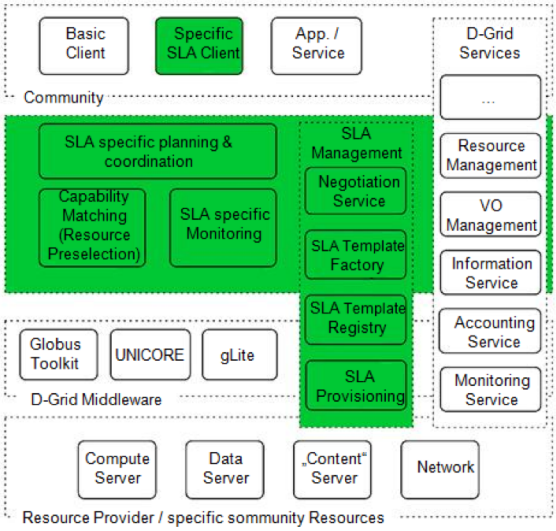


Figure 3.3.1: D-Grid SLA layer architecture [13]

The SLA4D-Grid [138] project developed an SLA layer for the German D-Grid infrastructure. The objective was to enable resource usage with a dedicated specified

[2]Or even is crucial for the performance depending on the type of parallelism of the distributed application.

QoS by the D-Grid communities and to increase reliability. A second objective was making the D-Grid open for new business models supporting commercial use.[3]

The SLA layer is seamlessly integrated in the D-Grid infrastructure and allows communities to use SLAs but does not enforce usage of SLAs. In the article "A Service Level Agreement Layer for the D-Grid Infrastructure" [13] we describe architecture and implementation realised in phase 1 of the project. Phase 2 was focused on developments left out in phase 1 in particular introducing negotiation in the D-Grid SLA and making additional term languages available. Figure 3.3.1 depicts the architecture, the components of the SLA layer are dyed green.

Besides SLAs for resource management we applied them for our novel approach for managing software licenses in distributed environments [31]. One goal using SLAs was to have a binding agreement between the license server and the user that allows to reserve licenses and features in advance for a certain time. Thus enabling pay-per-use business models also for environments where access to compute resources, e.g., HPC systems, for executing applications is not immediately but scheduled by a LRMS. In this way the license is not blocked early at the time of job submission as in the case of other software license management systems and can be used by someone else until the beginning of the reservation.

[3]However, it turned out that commercial use of the D-Grid resources at that time was not possible due the to public funding of resources in the participating computing centres, research centres and universities.

4. Term framework for Service Level Agreements

This chapter presents our work for the ***SLA terms*** component. The objective of the work is creating an integrated framework of SDTs (the OPTIMIS Service Manifest[1]) based on preliminary work on SDTs for different environments.

As described in the previous chapter a solution for creating SLAs should not be limited to a specific domain. Domain-specific information should not be part of the specification to keep it generally useful and interoperable across different domains. Hence, the WS-Agreement templates as defined in the specification need additional domain-specific terms to adapt the templates to an environment for which SLAs will be created. To support SLA negotiation between a user and several providers, e.g. to figure out which provider's offering fits best, all parties must share the same terms. Here, another level of standardisation is needed to achieve this interoperability between the user and different providers. Moreover, for hybrid Clouds this level of interoperability is also required between providers, and between a broker and different providers. Up to now, most of these approaches to standardised terms are only available as an outcome of best practice and not as the specification of an SDO. Related standardisation activities have been launched over the last years by different organisations, e.g., the TeleManagement Forum [150] , the National Institute of Standards and Technology [98], the Open Grid Forum [103], the Cloud Select Industry Group – Subgroup on Service Level Agreements (C-SIG SLA), and finally the ISO/IEC [71]. The current state of these efforts will be presented in Section 4.3 of this chapter.

Other approaches that define standardized languages to describe services, like the Topology and Orchestration Specification for Cloud Applications (TOSCA) [152] of the
Organization for the Advancement of Structured Information Standards (OASIS) [100], focus on the management of applications in Cloud environments rather than on the QoS of the applications. E.g., the TOSCA specification provides a language to

[1] see Annex A.2 for the complete schema

describe service components and their relationships for the portable semi-automatic creation and management of application layer services across alternative Cloud implementation environments so that the services remain interoperable. In this respect, TOSCA and the term framework described in this chapter are complementary since they support management and interoperability of services on different levels. When using TOSCA and the framework described in this thesis together, the TOSCA service templates would describe topology and orchestration of a service while the framework's SLA templates would support description and negotiation of the necessary QoS of the service to be achieved in the orchestration.

4.1 Early implementations of Service Description Terms

During the development of the WS-Agreement specification we studied in a number of projects several term languages with respect to their suitability for domain specific SLAs. We present the outcome in the next paragraphs.

In the VIOLA project we identified the OGF standard JSDL [75] as well suited candidate for describing properties of a computational job. For the co-allocation of network resource we extended the JSDL XML schema with terms needed to describe network QoS properties, e.g., bandwidth, delay, or jitter, that were processed by the ARGON network resource management system.

In the DGSI project [45] we used JSDL for activity delegation (job submission to remote community resources) for the same reason as in VIOLA and the OGF standard GLUE [56] for resource delegation. The rational for using the GLUE XML schema being the need for a term language to describe properties of the nodes to be delegated which cannot be achieved with the JSDL schema for the description of computational jobs.

In the SLA4D-Grid project we developed a generic XML schema for advance reservation that was used along with JSDL and GLUE.

4.2 OPTIMIS Service Manifest

In the context of the European project OPTIMIS we developed the most comprehensive XML schema (the OPTIMIS Service Manifest [127]). The goal was to provide a central schema for the definition of terms for the various aspects of SLAs in a Cloud environment with multiple infrastructure providers and multiple service providers. With the service manifest of the OPTIMIS project [109] we aim at providing a unified and extensible term language for optimising Cloud services in an IaaS Cloud environment.

Paper VI [50] provides a detailed discussion of the OPTIMIS project in Section 7.5.6. The OPTIMIS SLAs are based on WS-Agreement and follow its approach to negotiate and create SLAs on the basis of agreement templates. The manifest schema is designed to allow the Service Provider (SP) specifying requirements and constraints regarding the Cloud service of the Infrastructure Provider (IP). The Service Manifest consists of a section (the core), which is shared between the SP and IP and comprises several blocks. It has two additional sections (extensions)

that contain specific information for IP and SP. These extensions are not shared and only used in their respective domain.

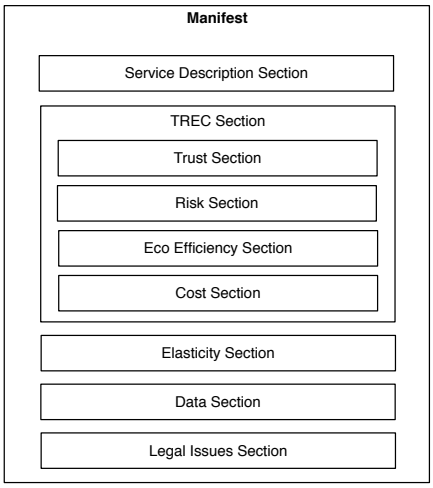


Figure 4.2.1: High level view of a manifest

Figure 4.2.1 presents a high-level overview of the core Service Manifest structure. The core is included in the agreement template and becomes part of the contract between SP and IP when the SLA is created. The extensions are not included in the template and consequently are not part of the contract concluded between SP and IP.

Figure 4.2.2 shows an example for a provider extension that defines infrastructure parameters required to deliver the Cloud service agreed upon with the service provider.

In the following paragraphs we briefly describe the components of the service manifest. Where available existing standards for expressing the terms of the different sections of the manifest have been used, e.g. several sections of the manifest include term descriptions based on Open Virtualization Format (OVF) [111] of the Distributed Management Task Force (DMTF) [47].

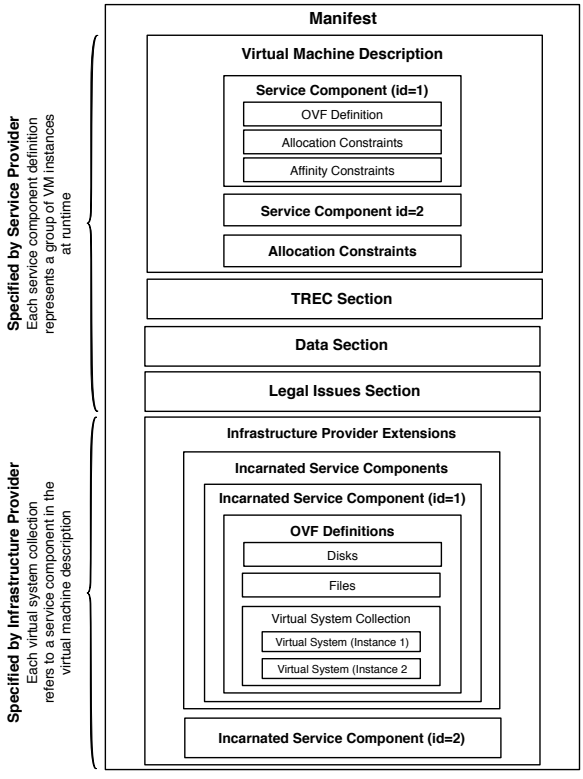


Figure 4.2.2: Example of an infrastructure provider extension in the manifest

Service Manifest global definitions: Used for housekeeping for the entire Service Manifest, e.g. setting the namespaces, define the way the Id of the Service Manifest) is built, determine the sequence of sections of the Service Manifest.

Service Description Section: A service may consist of several components. The Service Manifest used in OPTIMIS by default defines a virtual machine service component, that can be used to deploy a particular virtual system in the infrastructure of an IP.

Virtual Machine Description Section: Is a special instance of a *Service Description* which is used to describe one or more *Service Components* that together build a service and can be deployed.

Service Component Section: A *Service Component* has the following four blocks:

(i) OVF definitions: This is a template for creating instances of this component at the IP site. It provides information on location, format, network connections and the description of the virtual system to be used for creating component instances.

(ii) Allocation Constraints Section: This element is used to define initial, maximum and minimum numbers of component instances that can be created.

(iii) Affinity Section: Used by the SP to specify the level of affinity, i.e. the required proximity, between *Service Components*, which must be considered for deployment decisions., e.g. deploy in the same rack for maximum network performance.

(iv) Anti-Affinity Section: Allows the SP to specify the level of anti-affinity, i.e. the required distance between *Service Components* which must be considered for deployment decisions, e.g. do not deploy in the same data centre.

Trust, Risk, Eco-efficiency, Cost (TREC) Section: The TREC section contains the definition of the SP's requirements regarding trust, risk, eco-efficiency, and cost.

Trust Section: Can be used by the SP to define, e.g. the minimum level of trust required in case of IP driven Cloud federation.

Risk Section: Can be used by the SP to define, e.g. the maximum level of risk acceptable when using additional resource from another IP.

Eco Efficiency Section: Defines the required eco certification of an IP.

Cost Section: Can be used by the SP to negotiate, e.g. different price schemas with the IPs. Terms are adopted from Unified Service Description Language (USDL) [156] Pricing Module [155].

Elasticity Section: Describes the parameters and thresholds that trigger the *Elasticity Engine* to take corrective actions if the value of the associated KPI falls below a defined threshold or exceeds a threshold.

Data Protection and Data Security Section: Defines a list of countries according European legislation where data may be transferred to and a list of countries where data may not be moved to. This section also defines details how data have to be encrypted during transport over networks or when stored in the data centre.

Legal Issues Section: Besides aspects of functional and non-functional QoS properties OPTIMIS SLAs may contain legal terms such as Binding Corporate Rule, Standard Contractual Clause, or statements regarding the Intellectual Property Right.

In case of a Cloud bursting, multi-Cloud or Cloud federation scenario the manifest can be split to make part of the core available to other sites allowing to create their proper SLAs. However, only information of the shared section will be passed to other domains since they are not domain and provider specific. IP or SP components belonging to other domains are completely unaware of the existence of this information and will build and include their own extensions into the manifest according to their local conditions.

4.3 Current work on Cloud Service Description Terms

Since the end of the OPTIMIS project in 2013, a number of European-funded projects focussing on a common reference model for Cloud SLAs have been launched by the EC, e.g., SLA-Ready [140], SLALOM [141], SPECS [146]. While focussing on different aspects of Cloud SLAs none of these projects produced a machine-processable schema for creating SLAs. Rather their outcomes were guidelines, recommendations or legal documents concerning Cloud SLAs: SLA-Ready with a focus on support for security and data protection, SLALOM focussing on technical and legal models, and SPECS addressing security.

Additional SLA-related work to be mentioned includes results of activities of TMF, OGF, NIST, C-SIG SLA, and ISO/IEC. The 2014 TMF report *End-to-End Cloud SLA Management* [153] aims to "demystify the processes and methodologies to establish Cloud Service Level Agreement (SLA) and to identify interoperable standards required to facilitate end-to-end Cloud SLA management in a complex Cloud ecosystem" [153]. The OCCI working group [101] of the OGF has published a draft specification for the OCCI Service Level Agreements (OCCI SLAs) describing how the OCCI Core Model can be extended and used to implement a Service Level Agreement management API [102]. The proposed language is similar to WS-Agreement but differs in some definitions which makes it incompatible with the standard. NIST's 2014 publication *Cloud Computing Service Metrics Description* [99] presents an overview and discusses metrics (standard of measurement) of Cloud services to be used. This work has been included in the work of the ISO/IEC Joint Technical Committee for Information Technology (JTC 1)/Subcommittee (SC) 38 Cloud Computing and Distributed Platforms [71] working group 3. The C-SIG SLA has been set up in 2013 by the European Commission's Directorate General (DG) for Communications Networks, Content and Technology (CONNECT) to work on aspects of Cloud SLAs. The group has published *C-SIG - SLA standardisation guidelines fourth final draft* [40] in 2014 to provide a set of SLA standardisation guidelines for Cloud service providers and professional Cloud service customers. The work has been contributed to the ISO/IEC SC 38 working group 3. ISO/IEC JTC 1/SC 38 Cloud Computing and Distributed Platforms working group 3 published *Service Level Agreement (SLA) framework and terminology - Part 1: Overview and concepts* in 2014 [69]. The ISO working group has been working since 2014 on *Service Level Agreement (SLA) Framework - Part 2: Metrics* [70] which is expected to be released as draft by the end of 2016. It should be noted, however, that all of the efforts above do not address SDTs for Cloud services but were limited to preliminary work that may eventually lead to standardised service description terms. So far, developments of (mainly European) projects like OPTIMIS are still the only efforts towards the definition of generally applicable SDTs for Cloud SLAs.

Finally, in 2015 the EC published the results of a study on *Standards terms and performance criteria in service level agreements for Cloud computing services* [59]. While not addressing machine-processable SLAs in particular, the report provides a comprehensive overview on both technical and legal approaches used in practice of Cloud SLAs and includes suggestions for a model SLA. Many aspects of the OPTIMIS SLA can be found in this report again.

5. Electronic Negotiation of Service Level Agreements

This chapter addresses our work for the ***SLA negotiation*** component. In Section 5.1 of this chapter we first discuss negotiation models and early work to implement negotiation in WS-Agreement and other approaches. In Sections 5.2 and 5.3 we present the preliminary work approaching to a standard negotiation protocol and the work towards the final definition of WS-Agreement Negotiation.

WS-Agreement, domain-specific term languages and a protocol for electronic negotiation are three essential elements needed to create machine-processable SLAs. Negotiation and negotiation protocols in general have been under research for quite some time and we have seen increased research for negotiation protocols for SLAs in the last decade. The motivation behind being the fact that a simple request-response protocol as provided by WS-Agreement is not sufficient in many situations, e.g., when a composite service includes services from multiple providers, or when the optimal price of a service is can only be figured out by taking the offerings of multiple providers into account, or to determine the best service for a given requirement which has some flexibility, or when a broker acts on behalf of a user, or when co-allocating resources. The focus of our research was on developing a simple, robust though powerful protocol that supports existing standards. In particular, the protocol should extend WS-Agreement to allow multi-round negotiations without requiring changes to the specification of WS-Agreement.

5.1 Negotiation models

Negotiation models have been studied for quite some time. While many of the proposed approaches turned out to be isolated applications with little impact they contributed implicitly to the development of the standardised negotiation protocol that will be presented in Section 5.3. Controversial issues during the discussion of negotiation models for WS-Agreement were (i) the point in time when SLAs become binding for each of the two parties, and (ii) whether an SLA in force can be modified, e.g., by renegotiation. Most of the following approaches ignore the (legal)

implications of these relevant issues in a productive environment, both commercial and academic. Section 5.3 will return to these issues. The rest of this chapter presents some exemplary approaches taken since 2002.

5.1.1 Agent-based negotiations

One of the early standardisation approaches for negotiation has been specified for the agent domain by the Foundation for Intelligent Physical Agents (FIPA), which is a standards body for agent technology. Their initial specification was the Contract Net Protocol (CNP) [37], which defines a single round of bids limited by a deadline. Once a deadline has been reached, all bids are evaluated and the winners get the tasks. On top of the CNP FIPA has defined the Iterated Contract Net Protocol (ICNP) [72], which allows for a sequence of bids by one bidder. ICNP realises a protocol that supports multiple consecutive bidding rounds, which are limited by constraints defined by the auctioneer. Contract net has influenced a number of developments including OGF's WS-Agreement Negotiation.

Shen et al. [135] suggest using agents for negotiating optimised resource allocation for a job in a Grid environment. The agents can select between different negotiation protocols depending on, e.g., computation needs, available computing resources and their computing loads. Besides the Contract Net Protocol which the authors consider the most useful auction model, a model based on game theory, and a discrete optimal control model can be employed by the agents. The negotiations do not result in an SLA (proposal) but the job is passed to a job agent which selects the suitable resources, awards a contract to it and organises its execution. The fact that in this approach no SLA templates are used and in consequence no SLA (proposal) is created turns it incompatible with WS-Agreement. However, adapted negotiation protocols could be used in conjunction with WS-Agreement.

In "Service level negotiation in a heterogeneous telecommunication environment" [60] the author discusses a project with an industrial partner to explore the formulation and requirements of service level agreements in the domain of network operators, and the applicability of automated negotiation by agents. Using the Agent Communication Language (ACL) [4] developed by FIPA for agents negotiating the service levels the author considers as one of the promising approaches. The underlying FIPA developments have been considered for the development of WS-Agreement Negotiation.

In "Autonomous service level agreement negotiation for service composition provision" [169] Yan et al. propose agent-based agreement negotiation for a service composition. Within the proposed framework the service consumer is represented by a set of agents which negotiate quality of service constraints with the service providers for various services in the composition. The framework includes a negotiation protocol based on the FIPA ICNP to support coordinated negotiation. The coordination of the multiple negotiations is realised on top of the ICNP with an own approach that breaks down the composition level requirements into the service level requirements for individual services that will be negotiated by agents. In the second phase the negotiation coordination confirms or rejects the negotiation results. While the protocol is different from WS-Agreement Negotiation there is a similarity regarding the two phase process separating negotiation and creation of the SLA.

Hudert [67] considers the need for quality of service guarantees given the increasing role of continuously interacting highly specialised digital services and resources. He argues that "different market situations and negotiated products (i.e. SLAs) demand different negotiation protocols in order to reach the highest-possible overall efficiency of the system". The author presents an approach for developing a generic, service-based architecture as well as a set of protocols and data structures acting as a basic infrastructure for software agents to discover and negotiate about electronic services independent of the actual negotiation protocols applied. Similar to [28] the design proposal for a protocol-generic SLA discovery and negotiation infrastructure is using negotiation protocol description document(s) associated with a given service that can be retrieved by service users prior to starting an SLA negotiation employing software agent technology[1]. Followup work on the definition of a negotiation protocol description language has been published in [66]. The negotiation protocol description language can be used to define specialised service-specific negotiation protocols that could replace WS-Agreement Negotiation for WS-Agreement-based SLA negotiation. However, due to lack of standardisation interoperability between different negotiation parties is not provided.

Yaqub et al. [173] present a protocol development framework for SLA negotiations in Cloud computing. The framework is targeting agent-based negotiation and is based on a protocol development lifecycle, comprising four distinct phases namely modelling, verification, rule-based implementation and generic execution. The authors illustrate the framework by creating the Simple Bilateral Negotiation Protocol (SBNP) for multi-round negotiations in SaaS, PaaS and IaaS environments. The research has been conducted in the context of the SLA@SOI project [142]. The framework could also be used to generate an agent-based negotiation protocol for WS-Agreement since negotiation is decoupled from the agreement creation.

Yaqub et al. [171] address optimal negotiation of SLAs for Cloud-based services through agents. The paper addresses the SLA-gap existing between providers' preferences and customers' expectations which results in contemporary providers binding their offerings to the inflexible take-it-or-leave-it SLAs. The authors address this problem by presenting a negotiation strategy, which agents can use to create near-optimal SLAs under time constraints. Experimental evaluation is done against a variety of metrics including utility, social welfare, social utility and the Pareto-optimal bids. The development targets PaaS Cloud systems. The authors show that proposed negotiation of services may improve satisfaction of participants and reduce the SLA-gap. As the previously mentioned work the approach could be used together with WS-Agreement to replace the standard negotiation protocol.

5.1.2 WS-Agreement-based negotiation

Since first drafts of the WS-Agreement specification had been published a constant flow of developments could be observed that suggested extending the built-in one step request-response protocol negotiation of the specification.

Briquet and de Marneffe [26] present a survey of the current state and challenges of resource negotiation research, with a machine learning perspective. Regarding

[1] This approach is similar to our (negotiation) template mechanism in WS-Agreement and WS-Agreement Negotiation

negotiation of resource usage the authors see WS-Agreement as a relevant protocol development. For the sake of a stable negotiation protocol, machine learning techniques are considered by the authors as not applicable for the protocol part of resource negotiation.

In "A Negotiation Protocol Framework for WS-Agreement" [68] the authors define a meta-language for negotiation protocols that allow defining a multitude of specific negotiation protocols using a well-defined set of negotiation parameters. The approach takes into account that negotiation protocols cannot easily be incorporated in WS-Agreement without violating the standard. The core of the proposed approach is identical to the specification of WS-Agreement Negotiation as first the negotiation is performed and the resulting agreement proposal is then processed by WS-Agreement. The major difference regarding the negotiation protocol is that the meta-language allows for arbitrary negotiation protocols while WS-Agreement Negotiation incarnates just one standardised protocol.

Modica et al. [46] propose an extension of WS-Agreement to allow dynamic re-negotiations of SLAs. They argue that a committed WS-Agreement cannot be modified during service provision and is effective until all activities pertaining to it are finished or until one of the signing parties decides to terminate it. On the one hand the proposed extensions would be incompatible with the WS-Agreement specification. On the other hand the authors of the WS-Agreement Negotiation specification did not foresee re-negotiation of an agreement in force because of the contractual character of an SLA, i.e. re-negotiation takes place in the context of an existing contract. The proper way to modify an existing agreement includes the following steps: negotiate a new agreement, terminate the initial agreement and commit the new one to the state observed. This can be done both with WS-Agreement in one step or through a multi-round negotiation with WS-Agreement Negotiation. If dependencies exist between agreements as in the use-cases presented in the paper it would be up to one of the parties to synchronise the modifications of these agreements with a specific protocol on a layer above WS-Agreement.

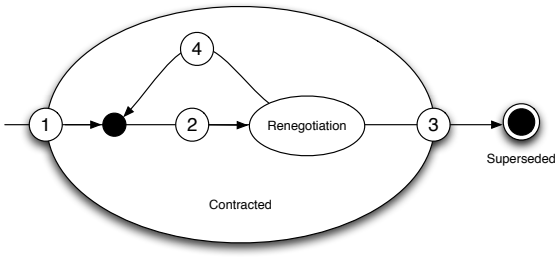


Figure 5.1.1: State machine of the contract re-negotiation protocol [115]

Parkin et al. [115] propose an SLA re-negotiation protocol. Their approach takes into account that an agreement in force is a contract and in consequence the re-negotiation takes place inside the *Contracted* state (*Observed* state in WS-Agreement) to reflect an existing contract is being re-negotiated. The authors define two additional states *Renegotiating* and *Superseded*. During re-negotiation the contract makes a transition to *Renegotiating* and transits to *Superseded* once the negotiation is successful, i.e. the new SLA will supersede the old SLA as depicted in Figure 5.1.1. Otherwise it returns to *Contracted*. The development was not continued after the end of the European projects that provided funding for the research. However, it had some impact on the development of WS-Agreement Negotiation although the GRAAP-WG decided not to include re-negotiation into the protocol but rather leave it as domain-specific specialisation on top of the generic WS-Agreement Negotiation protocol.

Micsik et al. [95] consider existing SLA specifications mainly address interoperability at a syntactic level. However, the terminology used can be very different and incompatible in terms of metrics and measurement units. The authors argue that ontologies and semantic technologies can address these problems as "ontologies can capture the meaning of the SLA related terminology in an unambiguous and machine understandable way, while semantic annotations within the SLA document can link any term to its well defined equivalent in the common ontology" [95]. The proposed approach includes an SLA Negotiator component working with existing SLA languages and protocols like WSLA or WS-Agreement. A prototypical development was done in the European BREIN project [25] but discontinued after the end of the project. However, some project members contributed results to the preliminary work for WS-Agreement Negotiation.

In "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility" [28] Buyya et al. present an implementation prototype of their Meta-Negotiation Middleware (MNM) that aims at bridging the gap between proprietary service interfaces and negotiation strategies of service providers and consumers. The meta-negotiation is based on a meta-negotiation document that may define the pre-requisites to be satisfied for a negotiation, such as a specific authentication method required or terms provider and consumer want to negotiate on (e.g. time, price, reliability); the negotiation protocols and document languages for the specification of SLAs (e.g. WSLA or WS-Agreement) that they support; and conditions for the establishment of an agreement, such as a required third-party arbitrator. Similar to the WS-Agreement templates published by the individual service providers MNM-using service providers publish their meta-negotiation document with descriptions and conditions of supported negotiation protocols in one or more public registries. Customers may use this registries for selecting a suitable provider and start SLA negotiations according to the conditions specified in the provider's document.

5.1.3 Other negotiation approaches

Meng and Brooke [93] argue that interoperability of service provision interfaces is important and though there have been many attempts to accomplish this by introducing uniform standards, dynamic negotiation for resource use and translation between interfaces provide a better alternative, given the dynamics of e-Science. The authors claim that existing negotiation protocol approaches are limited as they are based only on a Grid model and propose a new negotiation protocol for accessing arbitrary resources in an e-science collaboration environment. However, besides the initial publication no further related activities can be observed.

Besides developments like the work of Goiri et al. [57] that extend WS-Agreement in an incompatible way there are others that do not consider WS-Agreement for creating SLAs but develop their own framework for negotiating SLAs, e.g. the work of Chen et al. [33] that is proposing an agenda based approach to facilitate the multi-issue negotiation process between service consumer and service provider over the QoS requirements in Cloud services. Because the proposed negotiation approach is not agreement template-based it cannot be used to replace WS-Agreement Negotiation.

The European project SPECS [146] follows the WS-Agreement specification for creating SLAs and has developed their own negotiation protocol. Since the pro-

tocol was developed after the proposed WS-Agreement Negotiation standard their approach could not be influential for the standard. However, it may be used as an alternative negotiation protocol for WS-Agreement-based SLAs. Finally, SLA-Ready [140] and SLALOM [141], the other two European projects focussing on reference models for Cloud SLAs, are working on a more abstract layer that provides recommendations rather than proper definitions or implementations. SLA negotiation is not considered in their reference models.

5.2 Preliminary work on a standard negotiation protocol

"SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems" by Czajkowski et al. [41] was among the first approaches in distributed computing that aimed at solving the problem how to map activities such as computation or data transfer onto resources that meet requirements for performance, cost, security, or other quality of service metrics. The authors consider negotiation among application and resources to discover, reserve, acquire, configure and monitor resources. They argue that current resource management is limited in this respect and present a new approach defining a resource management model that distinguishes three kinds of resource-independent service level agreements (SLAs): Task Service Level Agreementss (TSLAs) defining tasks, Resource Service Level Agreementss (RSLAs) specifying resources, Binding Service Level Agreementss (BSLAs) that bind tasks to resources. The paper introduces states of the SLA, a resource and task meta-language, and defines a Service Negotiation and Acquisition Protocol (SNAP) for management of remote SLAs. SNAP was developed in the context of Grid computing, specifically for use on top of a preliminary version of OGF's Open Grid Services Architecture (OGSA). Core concepts of SNAP have influenced WS-Agreement and WS-Agreement Negotiation. However, SNAP was designed when OGSA was not yet developed for web service environments. As a result, WS-Agreement was designed using OASIS Web Services Resource Framework (WSRF) [165] for integration with the published specification of OGSA [104] that is also based on WSRF.

Riedel et al.[129] present their work on integrating WS-Agreement for resource coordination into the UNICORE [154] [148] Grid system. The implementation is a consequence of the analysis of the UNICORE system which identified the need for enhanced mechanisms to support both end-users and resource providers in the process of retrieving dynamic resource information and to negotiate on QoS guarantees. Moreover, the current UNICORE system did not include the means to manage resources based on economic models. The implementation realised an integration of the full WS-Agreement specification into UNICORE supporting the system's transition towards a service-oriented Grid with capabilities for standardised creation and negotiation of guarantees related to Grid services. Negotiation was not part of the implementation but kept in the responsibility of the respective Grid meta-scheduling systems instead, e.g. [10, 17, 78].

The paper "A Framework & Negotiation Protocol for Service Contracts" by Parkin et al. [113] presents "a specification for a domain-independent, symmetrical, two-party negotiation protocol to reach binding agreements between services

based on the principles of contract law". The authors argue "that the protocol is necessary as existing specifications, such as WS-Agreement, lack the capability to form negotiated agreements and lack sufficient rigour in their design" [113].

Their approach is based on international contract law as laid out in the United Nations (UN) Convention on Contracts for the International Sale of Goods (CISG) [133]. The major difference to many other proposed negotiation protocols is the clear definition that an offer for an agreement from the offerer is always binding and a contract is formed once the offerer receives the acceptance of the offeree. Contract formation is identical in WS-Agreement and the offer of the initiator (the agreement offerer in WS-Agreement) is obligating. The basic protocol is a simple request-response protocol similar to WS-Agreement. The simple agreement protocol is extended to a multi-round negotiation protocol by allowing counter offers from the party acting as offeree at that time. The WS-Agreement Negotiation protocol also stipulates a sequence of offers and counter-offers, the major differences being that WS-Agreement Negotiation has no contracted state (see Figure 5.3.3) since agreement creation is left to the *Agreement Layer* (see Figure 5.3.2).

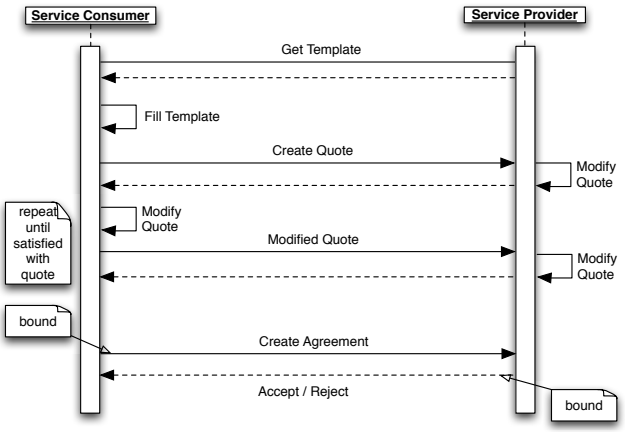


Figure 5.2.1: Sequence diagram of the negotiation process [175]

In "Extending WS-Agreement for dynamic negotiation of Service Level Agreements" [175] Ziegler, Wieder, and Battré summarise findings of recent work on SLA negotiation and their application in different use-cases. The report concludes with a proposal for a negotiation protocol for WS-Agreement depicted in Figure 5.2.1. The protocol is using the messages described in the contract renegotiation protocol [115]. Essentially, this approach supports both negotiation of new agreements and re-negotiation of existing agreements. However, additional conventions are requires to make sure that in case of re-negotiation the original agreement remains completely in force during the re-negotiation process.

Pichot et al. [119] propose and discuss extensions to the WS-Agreement protocol offering support dynamic negotiation and creation of SLAs. Considering QoS requirements of composed services co-allocation of resources, or distributed resource management in general the work motivates borrowing from the three phase commit protocol as developed for distributed databases. The paper presents a discussion and a first sketch of a negotiation protocol that will be the base of WS-Agreement Negotiation. It should be noted that although the negotiation state machine suggested in the paper looks similar to a state machine of a three phase commit protocol, the SLA negotiation and creation process is not a three phase commit. It is a blocking protocol as described by Skeen [137]. The proposed protocol already implied that negotiation must neither obligate the provider nor the consumer of the SLA as often one of the individual SLA might not be created due to the respective resource situa-

tion, e.g. co-allocation of resources requires agreements with more than on provider and not all providers might be ready to commit resources.

Quillinan et al. [123] address negotiation and monitoring of SLAs with a focus on actions that are applied when an agreement is violated. The paper discusses recent advances in this field and proposes some additional features that can help both consumers and producers during the enactment of services. These features include the ability to (re)negotiate penalties in an agreement, and specifically focus on the renegotiation of penalties during enactment to reflect ongoing violations. The authors argue that SLAs that include the definition of penalties which in turn require monitoring need a trusted third party, e.g. for trusted monitoring. They propose to include the TTP in the agreement, i.e. both parties agree on the TTP. Furthermore, for the request-response protocol of WS-Agreement they suggest splitting the negotiation template to allow separate negotiation of SLA, TTP, and penalties as negotiating either of them makes no sense if the respective other two cannot be successfully negotiated. Penalties in WS-Agreement SLAs lack flexibility when unexpected events interrupt enactment. The paper discusses the use of both multi-round negotiation and runtime re-negotiation of SLAs towards improving the experience for both service providers and consumers. The authors consider the current developments of the OGF's GRAAP-WG regarding SLA negotiation as promising environment for the implementation of the features presented in the paper.

5.3 Progress towards WS-Agreement Negotiation

Based on preceding work as described in Section 5.1 and the preliminary work described in Section 5.2 this section presents the final steps towards the specification of a multi-round negotiation protocol for WS-Agreement. The integration with WS-Agreement is realised following a loosely coupled layered design exploiting WS-Agreement's template mechanism.

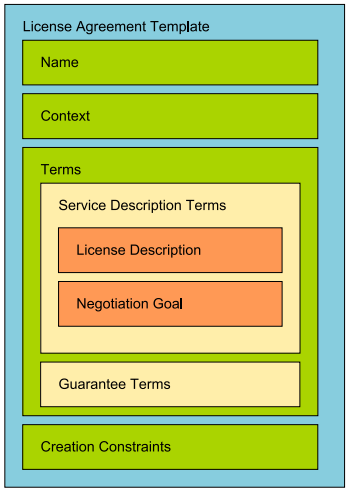


Figure 5.3.1: Structure of a SmartLM license agreement template [131]

While providing a first standard draft for a generic negotiation protocol the design allows to easily exchange this protocol with other, probably domain-specific protocols supporting, e.g, auction-like negotiations, or semantically enriched negotiations.

In "Extending WS-AGREEMENT with multi-round negotiation capability" [131] the state of OGF's work on the draft of the WS-Agreement Negotiation specification is presented along with a detailed discussion of the first implementation of the draft in the context of the European project SmartLM [145] [29]. For the implementation of WS-Agreement and WS-Agreement-Negotiation, the SmartLM component *SLA and Negotiation Service* uses the WS-Agreement Framework for Java [163]. WS-Agreement for Java (WSAG4J) implements both the basic features of the WS-Agreement protocol and the WS-Agreement Negotiation. The *SLA and Negotiation Service* implements the WSAG4J engine and exposes an agreement factory that provides an agreement template after the user requested a template (see Figure 5.3.1).

This template contains the XML schema of a license agreement including creation constraints. To retrieve the template, to negotiate the template and to create agreements, this agreement factory provides the following WSAG4J server actions:

GetLicenseTemplateAction implementing GetTemplateAction to retrieve a license template,

NegotiateLicenseTemplateAction implementing NegotiateAction to negotiate the variables of a license template according to the creation constraints,

CreateLicenseAgreementAction implementing CreateAgreementAction to finally create the agreement based on the initial template satisfying all creation constraints.

The user completes the *License Description* of the template with application name, and probably version and features of an application. The *Negotiation Goal* can be used to retrieve all features of an application with the corresponding maximum values. The corresponding negotiation goal is "FEATURES". The default negotiation goal is "TIMESLOT" which tells the server that the user wants to receive a free time-slot for his selected features and values. The resulting negotiation quote is sent to the *SLA and Negotiation Service* which in turn replies with a template filled with a valid license description and a free time-slot. This can be employed by the user to initiate the agreement creation or to continue negotiation with a modified template.

Paper I "A proposal for WS-Agreement Negotiation" [11] presents almost final details of WS-Agreement Negotiation to the scientific community. In this paper, the authors describe the Web Services Agreement Negotiation protocol, a proposal by the GRAAP-WG to extend the existing WS-Agreement specification: "This proposal is the result of combining various research activities that have been conducted to define protocols for negotiating service levels or to supersede the existing "take-it-or-leave-it" protocol. The main characteristics of this proposal are the multi-round negotiation capability, re-negotiation capability, and compliance with the original specification" [11] .

As described in Paper I WS-Agreement Negotiation offers three different negotiation models.

1. The basic Client–Server model, which is realised through asymmetric deployment of the WS-Agreement Negotiation Port Types.
2. The bilateral negotiation model, which is realised trough a symmetric deployment of WS-Agreement Negotiation, where the Negotiation Initiator is also the Agreement Initiator and the Negotiation Responder is the Agreement Responder. Both parties thus have an active role in the negotiation process.
3. The model for the re-negotiation of agreements. This is realised through symmetric signalling on the Negotiation and Agreement Layer. Both parties implement the WS-Agreement Negotiation and WS-Agreement port types. Moreover, both parties have their own instance of the original agreement. After the negotiation process, the responder of the original agreement creates the re-negotiated agreement.

The specification of WS-Agreement Negotiation was based on [11] with minor corrections and changes. Version 1.0 of the specification was published in 2011 as

proposed recommendation [158]. It has been implemented and used in a number of Grid and Cloud projects described in Chapter 6.

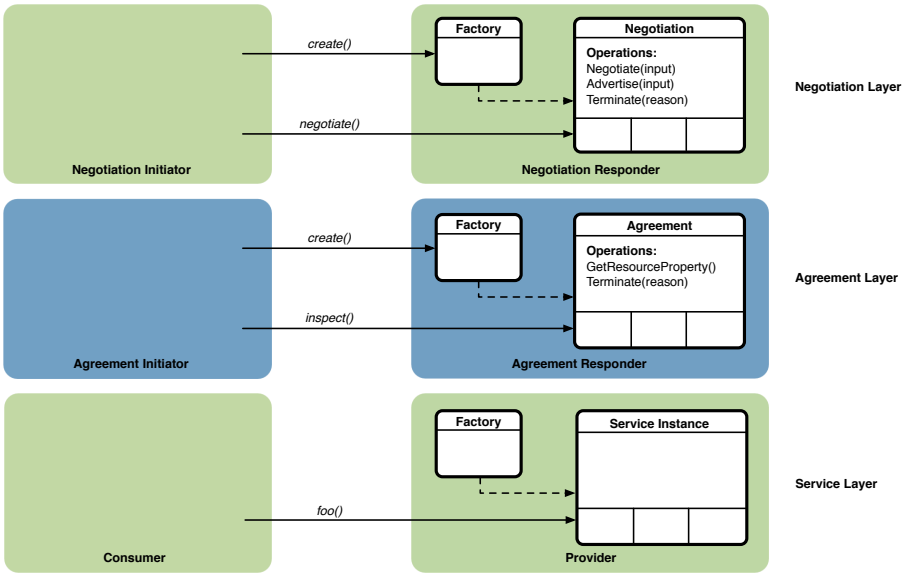


Figure 5.3.2: WS-Agreement Negotiation layered approach [158]

As discussed earlier, one important characteristic of the negotiation protocol is compatibility with WS-Agreement. Figure 5.3.2 shows the approach taken. Compatibility is achieved by adding a separate layer for the negotiation. The outcome of a negotiation (based on a negotiation template that is basically an agreement template plus negotiation constraints) is a valid agreement template that is then processed in the usual way in the agreement layer. It should be noted that successful negotiations do not imply that an SLA shall be created automatically. Rather, the outcome is an agreement offer that is binding for the negotiation initiator. The creation of the SLA, i.e. the commitment of the agreement responder is left to the discretion of the agreement responder as defined in the protocol of the agreement layer.

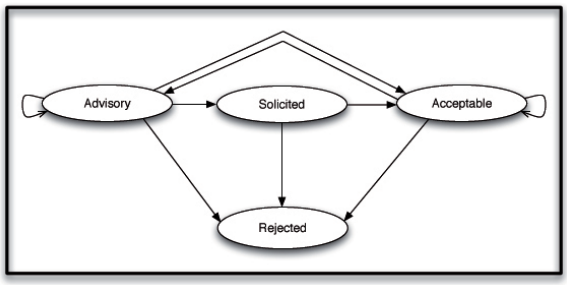


Figure 5.3.3: WS-Agreement Negotiation state machine [158]

Figure 5.3.3 depicts the state machine for the negotiation protocol. Messages exchanged during the negotiation are offers and counteroffers where each party may adapt the negotiation constraints for each offer or counteroffer. Also, each party may decide to start negotiation based on a new offer . Either party may decide that continuation of the negotiation is no longer expected to converge to an acceptable result and may enter the rejected state, which terminates the negotiation.

A Java reference implementation of WS-Agreement Negotiation is part of the WSAG4J framework that is available on sourceforge [163].

6. Service Level Agreements in Distributed Computing

This chapter present projects we have participated in and contributed to that have been used WS-Agreement and WS-Agreement Negotiation for the creation of SLAs in Grid and Cloud computing environments.

Machine-processable SLAs in the context of distributed computing infrastructures have been a topic of research and development work since around the year 2000. Most of the initial work was targeting Grid infrastructures managed by web-service-based Grid middleware. Some of this work with own contributions is presented in Section 6.1. Some other work is mentioned briefly. Section 6.2 does the same for Cloud computing. Most of the research and development work has been part of publicly funded projects. The report "Cloud Computing Service Level Agreements" [79] provides a comprehensive overview up to the year 2013. Since then, only a few new projects have emerged which have been briefly described in Section 4.3. However, working group 3 of ISO/IEC SC 38 on Cloud Computing and Distributed Platforms started addressing the standardisation of SLA terms in 2014 [71]. In Sections 6.1 and 6.2 we briefly present several Grid and Cloud projects projects to which we contributed SLA and resource management developments.

6.1 Service Level Agreements in Grid computing

According to the purpose of using SLAs in Grid projects they can be divided into two categories:

1. Co-allocation of resources (VIOLA [157] and PHOSPHORUS [117])
2. Activity or resource delegation (DGSI [45], SLA4D-Grid [138], IANOS [125])

WS-Agreement was the technology for SLAs we employed in all projects. We used JSDL and GLUE schemas (term languages) for describing QoS properties for computational jobs or computing resources to specify the SDTs of Grid SLAs, while we additionally developed a schema for advance reservation in SLA4D-Grid .

6.1.1 Co-allocation of resources

VIOLA was a project driven by the German National Research Network (NREN) Deutsches Forschungsnetz (DFN) and aimed on developing a solution for co-allocation of computing and network resources [10, 49]. The interest of the NREN was experimenting with bandwidth on demand in an optical network that was set-up as a testbed for future productive research networks. The developments in the project for the co-allocation of the two types of resources included a network management system for advance reservation of network links (ARGON [120]) between computing sites involved in the project and our meta scheduling service (MSS [160]) for the co-allocation. In VIOLA the MSS was loosely integrated into UNICORE [154] [148] to submit the job to the co-allocated computing resources after successful co-allocation (keeping them visible and manageable as Grid jobs in UNICORE).

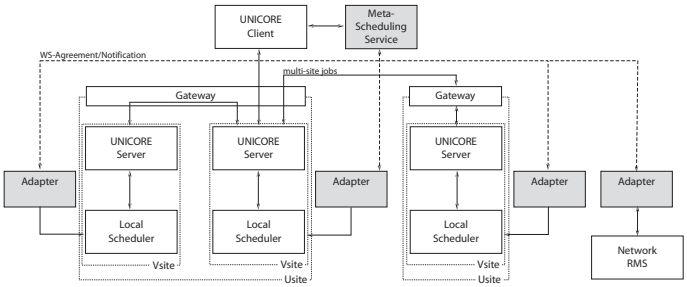


Figure 6.1.1: Architecture of the VIOLA meta scheduling environment [10]

The negotiation for the advance reservation was done by the MSS directly with the local schedulers of the computing sites through adapters, the main purpose of these adapters being the translation between the SLAs for advance reservation created by the MSS to the respective local resource management commands (the architecture is depicted in Figure 6.1.1). Similar, the MSS negotiated the advance reservation of the network links with the ARGON system through an adapter for the translation of the SLAs to the respective link reservations.

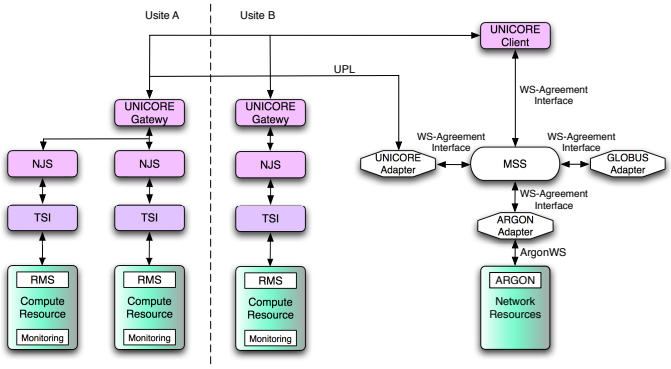


Figure 6.1.2: Initial architecture of the PHOSPHORUS meta scheduling environment

In the European project PHOSPHORUS [51, 117] we extended the VIOLA infrastructure to a European testbed (with several NRENs as operators) for bandwidth on demand in an all optical network. In the project we realised an improved integration of the MSS and UNICORE as shown in Figure 6.1.2. Now the MSS completely relies on the UNICORE system for managing the advance reservation

of computing resources which is realised through a UNICORE adapter that translates the advance reservation SLA to requests processed by the UNICORE Protocol Layer (UPL). Of course, this approach also requires extensions of the Target System Interface (TSI) to the local resources in oder to support the negotiation with the local Resource Management System (RMS) to achieve the advance reservation. Moreover, the UNICORE client was extended to transform the user request for a submitting distributed Grid job to the respective SLA for advance reservation.

In the last stage of the PHOSPHORUS project, the MSS did no longer directly communicate with a single network management system like ARGON but with several systems through an integration with the Network Service Plane (NSP) and the Control Plane (CP) respectively as required by the individual network environment. We realised the communication with adapters as depicted in Figure 6.1.3.

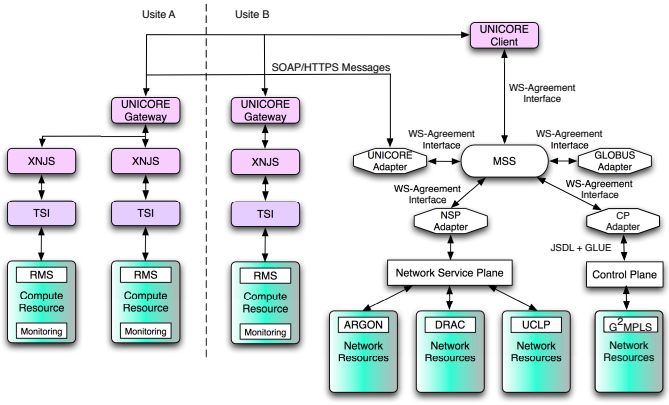


Figure 6.1.3: Final architecture of the PHOSPHORUS meta scheduling environment

6.1.2 Activity or resource delegation

The initial situation in the German Grid infrastructure D-Grid was characterised by resources predominantly contributed by different research communities around, e.g., earth science, particle physics, climate, or astronomy. For historical reasons each community had their own local resource management systems and a community scheduler that allowed job submission to these local RMS. With the communities joining D-Grid it became obvious that the different community schedulers didn't have a common interface to submit jobs to or to receive jobs from other communities. Hence, there were no generally usable D-Grid resources but a number of "incompatible islands" with resources. To develop a solution for this problem the BMBF-funded DGSI project was launched. In the project we envisaged two models for using resources of another community and identified both a common resource description language and an interoperability layer on top of the community schedulers as major prerequisites [19]. The two models for resource usage were activity delegation, i.e. job submission to remote communities, and resource delegation, i.e. a Cloud-like access to remote resource. We considered WS-Agreement-based SLAs for activity delegation and resource delegation as the way to communicate the requirements of both delegation models across the different community schedulers involved. As common resource description languages we employed JSDL for activity delegation and GLUE for resource delegation (both standards of the OGF). Paper VII "Infrastructure federation through virtualised delegation of resources and

services - DGSI: Adding interoperability to DCI meta schedulers" [18] provides a detailed presentation of the project.

In parallel to the DGSI project another BMBF-funded D-Grid project aimed at defining a general SLA layer in D-Grid: Service Level Agreements for D-Grid (SLA4G-Grid) [13]. While in DGSI we used SLAs to implement an interoperability layer for the community schedulers, in SLA4D-Grid we aimed at developing a generic SLA layer for the D-Grid which will enable communities to specify QoS requirements for resources either requested or provided. We designed the SLA4D-Grid SLA layer to integrate into the existing D-Grid architecture with the main objective to allow communities deciding the degree of integration with their local middleware. Figure 6.1.4 shows three exemplary communities of D-Grid and the different ways to use the SLA layer: plugged in between a community-specific middleware and the D-Grid middleware (right pillar), or users interacting directly with the SLA layer (middle pillar). Finally, communities that do not want to make use of SLAs just continue their way accessing the D-Grid middleware (left pillar).

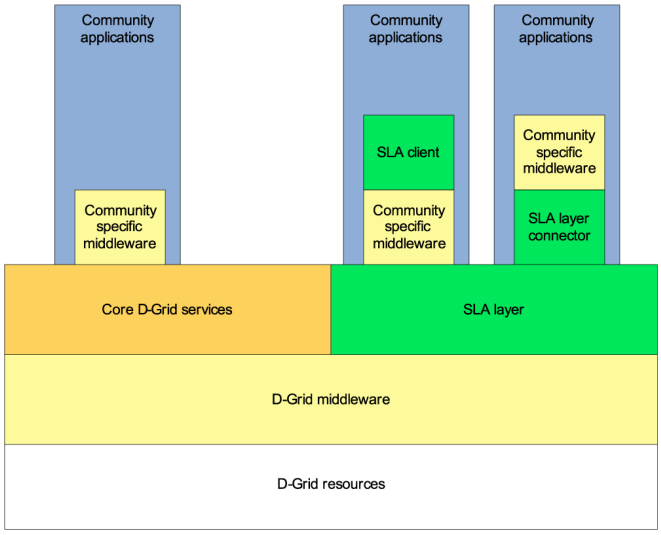


Figure 6.1.4: Integration of an SLA layer into D-Grid [13]

The project decided for WS-Agreement as base-technology for the implementation for several reasons: it is widely distributed, it is the only standard, it is intrinsically extensible to adopt to future requirements, it has an active developer community.

IANOS was a joint project of École polytechnique fédérale de Lausanne (EPFL) Lausanne, University of Applied Sciences and Arts of Western Switzerland (HES-SO) Fribourg, Swiss National Supercomputing Centre (CSCS) Manno, Technische Universität (TU) Dortmund, and Fraunhofer SCAI aiming at developing a framework for an application-oriented scheduling of HPC applications in a Grid of HPC computing resources. We designed a scheduling framework able to automatically select the optimal resources for executing an application with respect to execution time and cost. The basic idea was that applications could be characterised with respect to properties like, e.g., number of operations, number of messages sent, average size of messages. These properties can be determined both as estimation based on the type of application and from monitoring data from previous application executions.

For the HPC resources in the Grid, properties that influence the performance of an application have been gathered as well. Based on this data the framework is able to do a matching of application to best suited resource [39, 63, 78]. The architecture of the IANOS framework is presented in Figure 6.1.5. The Graphical User Interfaces (GUIs) are dyed red and the middleware components specific for IANOS are dyed green and blue. Below the green components is a standard UNICORE middleware environment complemented by IANOS-specific monitoring components.

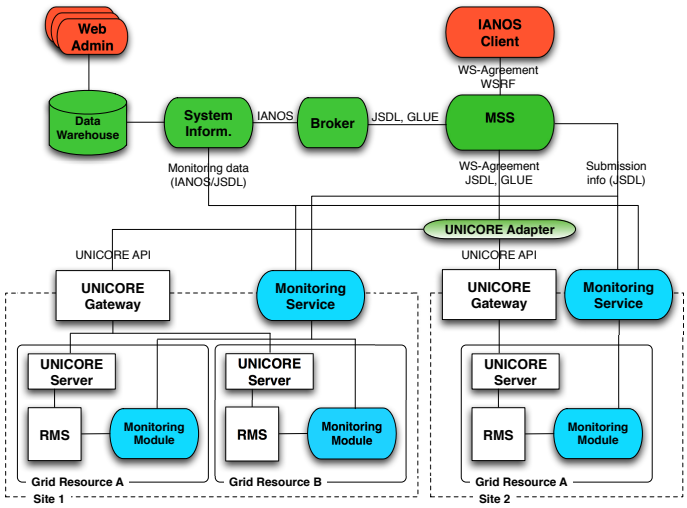


Figure 6.1.5: IANOS architecture [162]

For the framework we used WS-Agreement for creating agreement templates which capture the properties of the individual HPC resources. These template are published by the HPC sites and used during the MSS negotiating availability and cost of resources hosted by the different sites. Once the appropriate resource was found, an SLA was created between MSS and the middleware of the respective site including a reservation of the resources. JSDL and GLUE were used as description languages for resources and jobs. Figure 6.1.6 shows a high level view of the resource selection process.

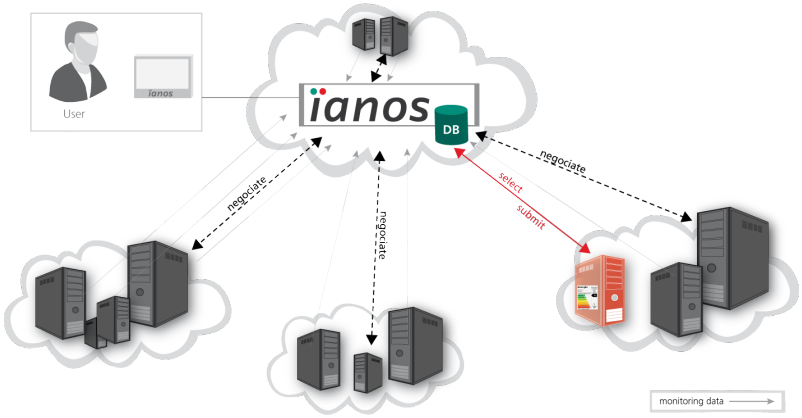


Figure 6.1.6: IANOS resource selection

6.2 Service Level Agreements in Cloud computing

Ever since Cloud resources have been used in productive environments as extension of or replacement for on-site resources, agreements on the quality of the provided service between provider and customer have moved into focus increasingly. In general, commercial Cloud providers' offerings of SLAs are concerning availability only while their customers would benefit from more detailed and significant SLAs to obtain the QoS their applications require.

The European-funded project OPTIMIS was starting from the observation that hybrid Clouds will be used more commonly (see Paper VI [50]). These hybrid Clouds are realised by private Clouds interacting with public and other Cloud providers. Figure 6.2.1 presents the different ways to combine resources to hybrid Clouds considered in OPTIMIS. In order to put them to good use, customers would need better control over functional and non-functional properties of the provided service, e.g, performance related metrics, data protection measures, eco-efficiency characteristics, deployment requirements and restrictions. Also, bursting from a private Cloud into a public Cloud or dynamically setting up a multi-Cloud environment to deploy an application should be controlled by SLAs [151]. It should be noted that only SPs and IPs make part of the OPTIMIS eco system. End-users (customers of SPs) are not considered in OPTIMIS.

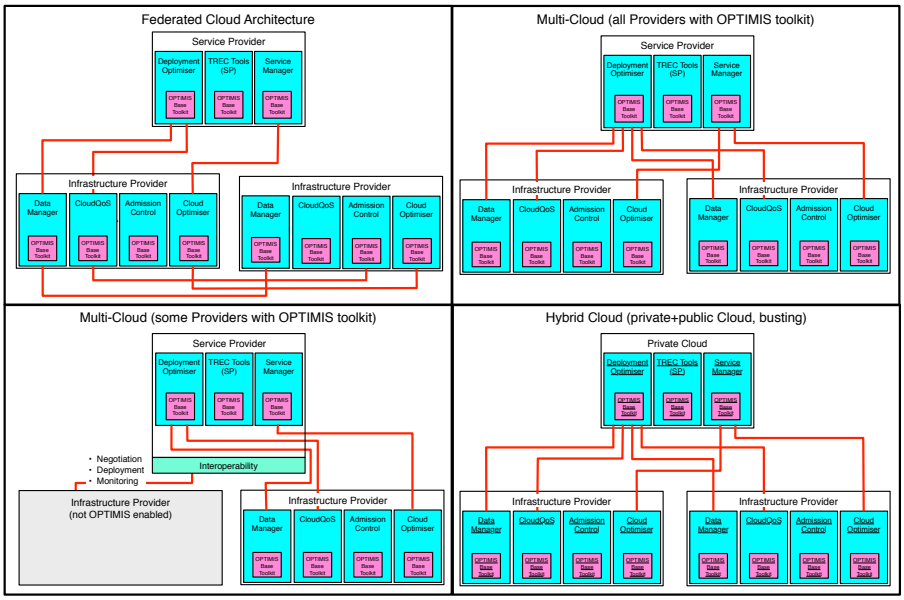


Figure 6.2.1: The OPTIMIS-supported Cloud architectures

In "Using Service Level Agreements for Optimising Cloud Infrastructure Services" [81] and "A Standards-Based Approach for Negotiating Service QoS with Cloud Infrastructure Providers" [126] we describe the implementation of our SLA component in OPTIMIS and the outcome of the initial work on the SDTs for OPTIMIS SLAs.

Figure 6.2.2 gives an overview on the components of the OPTIMIS toolkit and the flow of messages between them. The WSAG4J framework [163] for negotiating

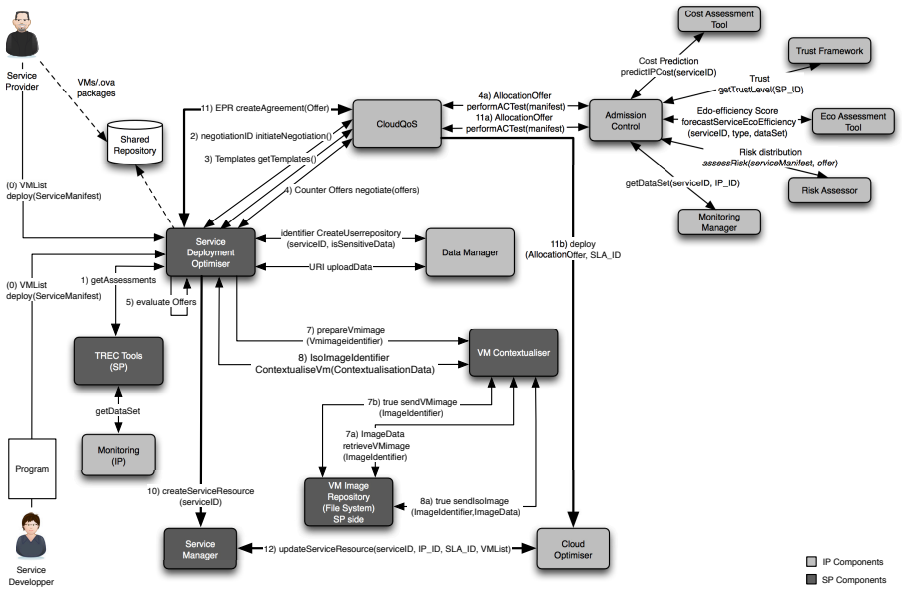


Figure 6.2.2: OPTIMIS toolkit

and creating SLAs is integrated in the toolkit as CloudQoS component (upper middle of Figure 6.2.2). For creating an SLA the SP's Service Deployment Optimizer component communicates with the CloudQoS (which is an IP component). To determine the feasibility of hosting the request of the SP the CloudQoS component queries the IP 's Admission Control component. Once the SLA has been created, the CloudQoS triggers the deployment through the IP's Cloud Optimizer component. Messages between the components are WS-Agreement templates instantiated as AllocationOffers containing the QoS requirements of the IP in form of a Service Manifest [127].

Parallel to OPTIMIS another European project was addressing Cloud federations: Contrail. Similar to OPTIMIS, Contrail is addressing federations of Clouds defined as a software infrastructure managing a set of independent IaaS providers that differentiate their services in the three dimensions of price, QoS, and Quality of Protection (QoP). Within the Contrail software infrastructure the provider also serves as an endpoint for SLA negotiations. User requirements are expressed in a uniform way, independently of the underlying resources. OVF is used to define the resources required for the application and the user starts a negotiation with these definitions [74]. The system will propose a price and the user may experiment by negotiating different configurations until the right balance between price and QoS is found (see Figure 6.2.3).

For negotiation and creation of SLAs, Contrail employs WS-Agreement together with SDTs for QoS and QoP defined. Like OPTIMIS, the SDTs include terms for specifying performance related characteristics of the deployment environment, data protection requirements or constraints or requirements for placement of Virtual Machines (VMs) when the service is deployed by the provider [32].

The earlier work of Lu et al. [85] done in the context of the SLA@SOI project also addressed SLA-based resource planning for multi-domain IaaS environments. The authors discuss the problem of planning resource outsourcing and local configurations for infrastructure services that are subject to SLAs. While OPTIMIS addresses

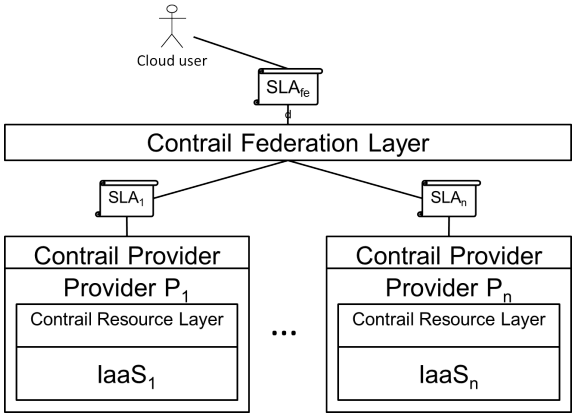


Figure 6.2.3: Contrail multi-level SLA negotiation [21]

the optimisation of TREC the approach presented is limited to minimising costs of implementation and outsourcing while respecting business policies for profit and risk. The approach implements a greedy algorithm for outsourcing, using cost and subcontractor reputation as selection criteria; and local resource configurations as a constraint satisfaction problem for acceptable profit and failure risks. Thus, it becomes possible to provide educated price quotes to customers and establish safe electronic contracts automatically. The approach is also suitable for modelling the specialised cases of infrastructure resellers and isolated infrastructure providers by discarding either local resource provisioning, or outsourcing, respectively.

More European Cloud projects with contributions to Cloud SLAs can be found in [79].

6.3 Grid and Cloud Broker

Finally, SLAs are used by brokers and meta-schedulers. Paper VII comprises a detailed description of our developments in the DGSI project to employ a meta-scheduler for brokering infrastructure federation through virtualised delegation of resources and services [18]. In "Connecting communities on the meta-scheduling level: the DGSI approach" [19] meta-schedulers of the D-Grid are presented along with the foundations of the negotiation-based delegation between meta-schedulers.

CompatibleOne is an Open Source development of a Cloud broker [170] which uses WS-Agreement for resource brokering and is available at [3]. Unlike many other developments that use the Java WSAG4J reference implementation as basis for the local WS-Agreement installation, CompatibleOne developed their own implementation of WS-Agreement using C. Agreement documents in CompatibleOne are created using an SLA template production tool which uses a combination of the standard WS Agreement and the CORDS service description model for service description specificities, placement conditions and business value guarantees.

7. Summary of Contributions

This chapter provides an overview on contributions made in the form of scientific publications and software artefacts. It further presents an estimation of the impact and concludes with short descriptions of the papers included in this thesis.

Contributions have been made to different aspects of Service Level Agreements in Grids and Clouds such as defining a framework of terms to describe QoS beyond availability, a language and a protocol to create SLAs, and a protocol for multi-round negotiation of SLAs.

7.1 Dynamic electronic Service Level Agreements

My major contribution for machine-processable SLAs was the work on the development of the WS-Agreement specification in the GRAAP-WG of the OGF . This included evaluation of WS-Agreement with specific term languages like JSDL with a number of use cases in different European projects as described for example in [14]. Paper IV presents work done in the context of a pre-commercial prototype development for a software license management system for using license-protected application in distributed computing. The core of the system for authorisation of license requests and licenses reservation is built around SLAs based on WS-Agreement. *elasticLM: A Novel Approach for Software Licensing in Distributed Computing Infrastructures* [29] presents results of the initial work for re-engineering the prototype towards the commercial product *elasticLM*®.

The impact of this work can be estimated by taking into account the broad adoption of WS-Agreement in various projects and developments, e.g., *elasticLM*®, cybula, CompatibleOne, or OCCI Agreement. There is a significant amount of published work related to WS-Agreement. A simple google search for "WS-Agreement" returns about 15.700 hits (including about 1.300 for WS-Agreement Negotiation) many of them linking to further research done in the context of WS-Agreement. WS-Agreement is the sole standard for the creation of machine-processable SLAs and on behalf of the OGF I am are currently preparing together with ISO/IEC JTC 1 its adoption it as an international standard.

7.2 Term framework for Service Level Agreements

Besides the technology for machine-processable SLAs appropriate term languages for the SDTs of the SLAs are needed. Here my major contribution has been the development of a prototype for advance reservation, preliminary work regarding energy efficiency terms for the OPTIMIS service manifest and further contributions to the service manifest, e.g., terms for data protection, or legal issues. The outcome of this work is presented in Paper II, Paper III and Paper IV. The complete OPTIMIS service manifest (term framework) is described in [127] and the schema is included in Annex A.2.

We would review the impact of the work based on the number of citations of the three papers according to google scholar: 2, 6, 7. Moreover, the OPTIMIS service manifest is the sole comprehensive term framework for both Cloud providers and their customers. It has been contributed to the work of the C-SIG SLA and the working group 3 of ISO/IEC's JTC 1. Additionally, it has been recently partly reused in OCCI.

7.3 Electronic Negotiation of Service Level Agreements

The extension of the simple request-response negotiation protocol of WS-Agreement was prepared in a number of preliminary research activities in the context of the OGF , described in, e.g. papers [115, 118, 119, 124, 131, 175]. My major contribution was the development of the specification of WS-Agreement Negotiation [158] in the GRAAP-WG of the OGF. The outcome of the work is described in Paper I [11].

The impact of this work can be estimated by taking into account the implementation in different projects and developments, e.g., OPTIMIS, *elasticLM*®. A simple google search for "WS-Agreement Negotiation" returns about 1.300 hits. WS-Agreement Negotiation is the sole standard for multi-round negotiations of machine-processable SLAs and as forWS-Agreement I am currently in negotiation with ISO/IEC JTC 1 to adopt it as an international standard. According to google scholar the number of citations of paper I is 20.

7.4 Service Level Agreements in distributed computing

Here the approach was introducing developments made around WS-Agreement and WS-Agreement Negotiation into projects to evaluate these developments in different environments and use-cases. Since some of the projects were ongoing during the development of WS-Agreement and WS-Agreement Negotiation valuable feedback from the respective implementations could be gathered.

7.4.1 Service Level Agreements in Grid computing

My contributions concerned the integration of WS-Agreement and WS-Agreement Negotiation in the Grid projects VIOLA [10, 49], DGSI [19], SLA4D-Grid [13], PHOSPHORUS [51]. An evaluation of ongoing Grid projects regarding the use of

SLAs for resource management was published as survey [134]. The DGSI work is described in more detail in Paper VII. Google scholar reports the number of citations as 6.

7.4.2 Service Level Agreements in Cloud computing

My contributions to the OPTIMIS project [151] were focussed on the development of the SLA management and the development of the Service Manifest as leader of the responsible work package. After the end of the OPTIMIS project, our developments have been contributed to the European Cloud middleware OpenNebula [108]. The project is described in more detail in Paper V and Paper VI. According to google scholar the number of citations is 2 and 272, respectively.

7.5 Paper overview

The papers included in this thesis study machine-processable SLAs from different angles: negotiation of SLAs, SLA-based resource management, and term languages for SDTs, and present other software artefacts that have been developed in this context. In paper I we study the foundations and a protocol for the negotiation of SLAs between different software-represented parties of an SLA. These could be, e.g., web interfaces of a web service back-end, or agents. Different aspects of the preparatory work leading to the complete term framework for SDTs developed in the context of the European OPTIMIS project are discussed in papers II and III. Three papers (V, VI, VII) investigate different aspects of employing SLAs for resource management. Cloud SLAs are discussed in papers V and VI, while paper VII presents our contributions to using SLAs in Grids for resource management in an academic environment. Paper IV studies the application of SLAs in the specific business segment technology for software licensing and protection of intellectual property and presents the work done in the context of the SmartLM project.

7.5.1 Paper I

Paper I [11] investigates SLAs and their negotiation based on a use case where provisioning of services for a user shall happen at pre-determined fixed times, which the provider assures through SLAs for negotiated advance reservation of resources. The negotiation model developed is based on WS-Agreement [158] templates which expose the offerings of the provider in terms of, e.g., QoS, availability, or service classes. From these templates the user selects one to start the negotiation. The paper describes the SLA negotiation model, which follows a three-tier approach with the negotiation layer on top of the agreement layer, which in turn is positioned above the service layer. Hence, the negotiation layer can easily be exchanged by another one with different capabilities if needed. The negotiation protocol supports bilateral negotiations with offers and counter-offers and provides full symmetry regarding rights, obligations and capabilities of provider and user. The paper describes the structure of an offer as well as the negotiation offer states. Negotiation factory and negotiation port types are illustrated through three representative negotiation examples: a basic client-server negotiation, a bilateral negotiation with asymmetric agreement layer and the re-negotiation of agreements with a symmetric agreement layer.

My contributions to this paper were based on my work for developing preliminary versions of the proposed negotiation protocol, e.g. [119, 131] and on the work as co-chair in OGF's GRAAP-WG that resulted in the specification of WS-Agreement Negotiation [158].

7.5.2 Paper II

Paper II [174] studies the development of a term language for the specification of user requirements regarding energy efficiency of the provider with an introductory overview on development and standardisation work for Cloud SLAs. The development is carried out in the context of the European project OPTIMIS [109] and extends our previous work on this subject [81]. The term language implements three approaches for specifying energy efficiency properties of a provider: (i) based on static certification information available up-front, (ii) based on dynamic data regarding average of energy consumption and carbon footprint gathered during a previous period and (iii) based on the availability of real-time monitoring data of actual power consumption. Approaches (i) and (ii) are useful for automated provider selection. The accuracy of the data depends in (i) on the quality of the certifying body and the age of the certificate, while in (ii) the point in time and the length of the measurement influence the accuracy. Approach (iii) allows continuous evaluation of the SLOs to determine compliance of the SLA, while in case of (i) and (ii) the corresponding SLOs are evaluated only once when the agreement is created. Moreover, the paper presents examples for the implementation of the SLOs as part of the term framework.

My contribution to the work presented in this paper was the development of the schema for SDTs regarding service providers' requests for eco-efficiency and eco-efficiency properties of Cloud providers' data centres.

7.5.3 Paper III

Paper III [9] examines legal aspects and security requirements regarding processing personal data in the Cloud (as defined in the European Data Protection Directive (DPD)) and how these can be implemented through SLAs. We analyse the legal conditions for data transfers and storage in Clouds outside the European Economic Area (EEA). Legal requirements are considered being part of the QoS a Cloud user requires from its Cloud provider. The detailed analysis is used to propose a technical framework that can be used by infrastructure providers to implement the requirements identified in the analysis. The baseline technology for the implementation for negotiating and creating the SLA is WS-Agreement and WS-Agreement Negotiation. For the definition of restraints and requirements regarding data placement and data protection in SDTs and SLOs we use a schema based on the OVF standard [112] of the DMTF [47]. The implementation in the OPTIMIS toolkit includes a location-aware data management system, and application level data encryption on top of storage device encryption during transport. The distributed Cloud file system prohibits placement of data outside the territorial scope of the Data Protection Directive (DPD) unless additional legal safeguards like SCC and BCR are agreed upon in the SLA. Please note: The analysis of legal requirements in this paper is based on the DPD of 1995 (Directive 95/46/EC), which has been repealed by the Data Protection Regulation (REGULATION (EU) 2016/679) April 2016 [44]. However,

while the new regulation became effective on 24 May 2016 it shall apply (i.e. shall be implemented in national law by the member states) from 25 May 2018 on only.

My contributions to the work presented in this paper was the integration of legal issues in the OPTIMIS SLA based on intensive cooperation with Universität Hannover being the responsible partner for legal issues in the OPTIMIS project and my work on the Service Manifest in OPTIMIS and term languages for SLAs, like in [21, 174] or in the developments of the SmartLM project [31].

7.5.4 Paper IV

Paper IV [31] is a comprehensive study on our novel approach for managing software licenses. The resulting license management technology has been developed in the European project SmartLM. The paper analyses the obstacles that hamper the use of commercial software in Grids and Clouds and presents an approach to overcome these obstacles: (i) decoupling the authorisation for using a license from the authorisation for executing a license protected application, (ii) making obsolete the communication with a license server hosted at the user's premises at runtime, (iii) providing a negotiated SLA between user and license server on availability of a license and corresponding features at a given time, for a defined period, and for dedicated computing resources, (iv) generating a software token that carries all information allowing the application API to validate the authorisation, (v) implementing sophisticated security features to protect the software license token against fraud and to allow the API to validate the token prior to the execution of the application. The paper describes the architecture, the SLA integration, the implementation of security, the evaluation of the SmartLM components, and different usage scenarios. In parallel to the technical developments new business models for Independent Software Vendor (ISV) in distributed computing environments have been developed and are presented in the paper.

My contributions to the work presented in this paper were the definition of the overall architecture of the SmartLM framework and the integration of WS-Agreement. Moreover, my contributions were based on contributions to other publication regarding the project, e.g. [30, 65, 83], and the fact that I was the scientific coordinator of the SmartLM project. As scientific coordinator I was responsible for defining and structuring the overall architecture, software developments and their implementation and integration required to achieve the targeted innovations. I also filed an application for a patent for the licensing framework.

7.5.5 Paper V

Paper V [21] investigates application and benefits of SLAs for creating Cloud federations. The research results presented are developed in the context of two European funded projects: Contrail [2, 38] and OPTIMIS [110, 151]. Both projects address creation of federations of Clouds through negotiation of SLAs based on templates offered by the providers. In contrast, previous approaches were based on information services, which we consider both less accurate (tending to be outdated over time) and providing less information on the provider resources regarding, e.g. capabilities. Federations allow combining resources of multiple Cloud providers to obtain the desired QoS which cannot be delivered by a single provider. The paper

presents how Contrail uses OVF for describing the virtual machine templates and WS-Agreement for defining QoS and QoP, e.g. for assuring the required geographical distribution of Cloud systems. In addition to Contrail's user driven federation, OPTIMIS supports provider initiated federations, similarly based on provider templates and using agreements for QoS. Some examples for terms to specify SDTs in agreement templates and SLOs in agreements developed in the two projects are discussed.

My contributions to the work presented in this paper were based on my work on the Service Manifest in OPTIMIS and individual term languages for SLAs.

7.5.6 Paper VI

Paper VI [50] studies fundamental challenges in Cloud service provisioning and provides solutions for (i) service life cycle optimisation, (ii) dependable sociability (= trust + risk + eco + cost), (iii) adaptive self-preservation, (iv) multi-Cloud architectures, and (v) market and legislative issues. The stakeholders in this work are IPs and SPs, the latter offering services to their users based on hardware resources provided by IPs. End-users as customers of an SP are not targeted in OPTIMIS. SLAs and their negotiation are an important part of service deployment and service execution (building the three basic steps of the service life cycle together with service construction). The outcome of the OPTIMIS research carried out in response to the challenges is a toolkit with components for realising a variety of architectures for simultaneous use of multiple Clouds. The toolkit comprises distinct components for the SP and IP as well as components that are used by both, e.g., the SLA manager. To cover all aspects of IP capabilities and SP requirements in the SLA a comprehensive schema (the OPTIMIS service manifest) has been developed with an API that allows to dynamically create an adapted agreement template for the negotiations of QoS between SP and IP, or IP and IP. Besides these standard scenarios, components of the toolkit also may be used by Cloud brokers thus allowing negotiations between SP and broker and broker and IP(s) to organise a federation.

My contributions to the work presented in this paper (primarily developing the overall architecture for integration of WS-Agreement and WS-Agreement Negotiation) were rooted in my responsibility as work package leader for the work package on SLAs and the work package on software licensing as well as my contributions to other OPTIMIS-related publications, e.g. [81, 126, 174]. As work package leader I defined and structured the software to be developed and guided their implementation.

7.5.7 Paper VII

Paper VII [18] presents extensive research and development for enhancing quasi-static Grid environments with Cloud-like elasticity. The work has been done in the framework of the German D-Grid project DGSI [45]. The objective of the project is developing an interoperability layer for the heterogeneous D-Grid community schedulers (meta-schedulers for the local RMS providing access to different resources of the individual communities) that supports both activity delegation (job submission to resources of another community) and resource delegation (making resources form another community temporarily available for local use). The interoperability layer is

realised through enabling the community schedulers to negotiate activity delegation or resource delegation with other community schedulers. For the negotiation we use WS-Agreement and a preliminary version of WS-Agreement Negotiation. For describing SDTs and SLOs we use two different standards that best fit for the respective purpose: (JSDL) [75] for activity delegation and GLUE 2 [56] for resource delegation. GLUE 2 is an XML schema for an information model for Grid entities. Agreement-based activity delegation makes straight forward use of the existing local facilities of the community. The only prerequisite are community schedulers supporting WS-Agreement SLA negotiation. Resource delegation includes virtualisation and temporary access to community resources. Thus, additional effort is needed to address issues like trust and security for virtual front-end systems, firewalls, naming administration, and more. This is handled by the Delegation Daemon (DD) running at each community. The DD exposes the interfaces for SLA management, file staging, VM storage, provisioning, control and is in charge of transforming the SLA received for the community scheduler into the requested virtual environment deployed into the delegated resources.

My contributions to this paper (mainly concerning the integration of WS-Agreement and WS-Agreement Negotiation and the architectural prerequisites) were based on my work on the specifications of WS-Agreement and WS-Agreement Negotiation [6, 158] as co-chair of OGF's GRAAP-WG and related publications like [118, 119, 131].

8. Conclusions

We conclude the thesis with a summary of the findings of our work and an outlook regarding future research and development in the area of the thesis.

This thesis demonstrates relevance and application of SLAs for managing QoS in distributed computing, in particular Cloud computing. It presents and summarises parts of my research work during the last decade resulting in a framework of three components (***SLA creation***, ***SLA terms***, and ***SLA negotiation***) that provide the technological base for the SLA-based QoS management. The thesis includes an investigation into the necessity of and requirements for the three components, presents related research and the approaches and efforts in various European and German projects for implementing the components of the framework.

In particular,

- the ***SLA creation*** component makes available the domain-agnostic language and a simple request-response protocol for creating SLAs. Developments - notably WS-Agreement - and related work have been described in Chapter 3 and papers VII and IV studies integration and use of WS-Agreement in a Grid computing environment where activity and resource delegation is governed though SLAs, and in the software license management framework *elasticLM*®. Paper VI presents the OPTIMIS toolkit (developed for IaaS Cloud environments) with its CloudQoS component that implements WS-Agreement for the creation of SLAs.

- Chapter 4 presented our developments and related work regarding the ***SLA terms*** component that provides the definitions of the SDTs allowing to use the ***SLA creation*** component for the creation of SLAs in different environments, namely SDTs for energy efficiency, software licenses, Cloud federation and legal information concerning data protection: BCRs, SCCs and IPRs. The related papers II, III, IV, V, and paper VI study SDTs for different environments and show the evolution of the SDTs from preliminary term sets and their application specific domains towards the comprehensive OPTIMIS Service Manifest.

- In Chapter 5 we described the work for the ***SLA negotiation*** component that extends the ***SLA creation*** component with the protocol to execute multi-round negotiations. Paper I investigates the protocol and its integration with WS-Agreement and presents an explanatory use-case for the application of SLA negotiation. Paper VI presents the OPTIMIS toolkit where WS-Agreement Negotiation is used by the OPTIMIS CloudQoS component for negotiating complex SLAs between SPs and IPs.

Although this thesis includes results of research and developments carried out over almost a decade there is room for research and developments to further enhance the outcome and increase the impact. We see future work in three areas: negotiation protocols, SDTs, and integration in today's web service environments. Regarding the negotiation protocols we consider beneficial the integration and evaluation of different negotiation protocols with WS-Agreement, in particular agent-based negotiation approaches, with respect to the growing number of machine-based Cloud resource brokering approaches. In the area of SDTs there is a need to integrate the - so far unconnected - different SDT developments into a comprehensive standardised set of terms. Finally, accessing web services is nowadays often no longer based on the SOAP stack but on RESTful technologies. Here we see additional effort required for studying and realising RESTful implementations of the WS-Agreement standard.

References

[1] SLA Management Handbook - Enterprise Perspective. Technical Report GB 917-4, 2004.

[2] Contrail White Paper. Technical report, Contrail, November 2012. Online at http://contrail-project.eu/documents/18553/341152/WhitePaper.pdf, visited 30 June 2016.

[3] CompatibleOne ACCORDS Platform Repository. Website. Online at http://gitorious.ow2.org/ow2-compatibleone/accords-platform, visited 10 June 2016.

[4] FIPA Technical Committee Communication. FIPA Agent Communication Language Specifications, 2002. Online at http://www.fipa.org/repository/aclspecs.html, visited 31 May 2016.

[5] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement) proposed recommendation. Technical Report GFD.107, May 2007.

[6] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). Technical Report GFD.192, October 2011.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical report, February 2009.

[8] Lee Badger, Tim Grance, Robert Patt-Corner, and Jeff Voas. NIST Cloud Computing Synopsis and Recommendations. Technical report, May 2012.

[9] Benno Barnitzke, Wolfgang Ziegler, George Vafiadis, Srijith Nair, George Kousiouris, Marcelo Corrales, Oliver Wäldrich, Nikolaus Forgó, and Theodora Varvarigou. Legal Restraints and Security Requirements on Personal Data and Their Technical Implementation in Clouds. In *eChallenges e-2011. Proceedings.*, 2011.

[10] Christoph Barz, Thomas Eickermann, Markus Pilz Pilz, Oliver Wäldrich, Lidia Westphal, and Wolfgang Ziegler. Co-Allocating Compute and Network resources - Bandwidth on Demand in the VIOLA Testbed. In *Towards Next Generation Grids, Proceedings of the CoreGRID Symposium, Rennes, France August 2007, Springer CoreGRID Series, Voume 5*, pages 193–202, 2007.

[11] Dominic Battré, Frances MT Brazier, Kassidy P Clark, Michael Oey, Alexander Papaspyrou, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. A proposal for WS-Agreement Negotiation. In *Proceedings of the IEEE Grid 2010 conference, October 5 - 28 2010, Brussels*, October 2010.

[12] Dominic Battré, Odej Kao, and Kerstin Voss. Implementing WS-Agreement in a Globus Toolkit 4.0 Environment. In Domenico Talia, Ramin Yahyapour, and Wolfgang Ziegler, editors, *Grid Middleware and Services: Challenges and Solutions*, pages 409–418. Springer CoreGRID Series, Volume 8, 2007.

[13] Dominic Battré, Roland Kübert, Axel Tenschert, Oliver Wäldrich, and Wolfgang Ziegler. A service level agreement layer for the D-Grid infrastructure. In *eChallenges 2010*, 2010.

[14] Dominic Battré, Philipp Wieder, and Wolfgang Ziegler. WS-Agreement Specification Version 1.0 Experience Document. Technical Report GFD.167, 2010.

[15] Claude Baudoin, Jordan Flynn, John McDonald, John Meegan, and Michael Salsburg. Public Cloud Service Agreements: What to Expect and What to Negotiate. Technical report, April 2013. Online at http://www.cloud-council.org/publiccloudSLA.pdf, visited 13 May 2016.

[16] Berkeley, CA, USA. *Defining and Monitoring Service-Level Agreements for Dynamic e-Business*, 2002.

[17] Boris Bierbaum, Carsten Clauss, Thomas Eickermann, Lidia Kirtchakova, Arnold Krechel, Stephan Springstubbe, Oliver Wäldrich, and Wolfgang Ziegler. Reliable Orchestration of distributed MPI-Applications in a UNICORE-based Grid with MetaMPICH and MetaScheduling. In *Recent advances in parallel virtual machine and message passing interface : 13th European PVM, MPI User's Group Meeting, Bonn, Germany, September 17-20, 2006; Proceedings Berlin: Springer, 2006 (Lecture Notes in Computer Science*, pages 174–183, September 2006.

[18] Georg Birkenheuer, André Brinkmann, Mikael Högqvist, Alexander Papaspyrou, Bernhard Schott, Dietmar Sommerfeld, and Wolfgang Ziegler. Infrastructure Federation Through Virtualized Delegation of Resources and Services. *Journal of Grid Computing*, 9(3):355–377, July 2011.

[19] Georg Birkenheuer, Arthur Carlson, Alexander Fölling, Mikael Högqvist, Andreas Hoheisel, Alexander Papaspyrou, Klaus Rieger, Bernhard Schott, and Wolfgang Ziegler. Connecting communities on the meta-scheduling level: the DGSI approach. In *Cracow Grid Workshop '09*, pages 96–103, 2010.

[20] C Bitten, J Gehring, U Schwiegelshohn, and R Yahyapour. The NRW-Metacomputer - building blocks for a worldwide computational grid. In *Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th*, pages 31–40, 2000.

[21] Lorenzo Blasi, Jens Jensen, and Wolfgang Ziegler. Expressing Quality of Service and Protection Using Federation-Level Service Level Agreement. In Dieter an Mey, Michael Alexander, Paolo Bientinesi, Mario Cannataro, Carsten

Clauss, Alexandru Costan, Gabor Kecskemeti, Christine Morin, Laura Ricci, Julio Sahuquillo, Martin Schulz, Vittorio Scarano, Stephen L Scott, and Josef Weidendorfer, editors, *Euro-Par 2013: Parallel Processing Workshops*, pages 146–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[22] Florian Blümel, Thijs Metsch, and Alexander Papaspyrou. A RESTful Approach to Service Level Agreements for Cloud Environments. In *2011 IEEE 9th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 650–657. IEEE.

[23] BPEL4WS - Business Process Execution Language for Web Services. Website. Online at https://msdn.microsoft.com/en-us/library/ee251592%28v=bts.10%29.aspx, visited 10 May 2016.

[24] Greg Brain, David Banes, Debbie Burkett, Kelly Flynn-Muller, Jane Hall, Malcolm Sinton, Carolyn Smithson, and Tobey Trygar. SLA Management Handbook - Concepts and Principles. Technical Report GB 917-2, 2005.

[25] BREIN project flyer. Website. Online at http://cordis.europa.eu/pub/ist/docs/grids/brein_en.pdf, visited 31 May 2016.

[26] Cyril Briquet and Pierre-Arnoul de Marneffe. Grid resource negotiation: Survey with a machine learning perspective. In *Proceedings of the 24th IASTED International Conference on Parallel and Distributed Computing and Networks*, PDCN'06, pages 17–22, Anaheim, CA, USA, 2006. ACTA Press.

[27] Maria Grazia Buscemi and Ugo Montanari. CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements. In *Programming Languages and Systems*, pages 18–32. Springer Berlin Heidelberg, Berlin, Heidelberg, March 2007.

[28] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James 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*, 25(6):599–616, 2009.

[29] Claudio Cacciari, Francesco D'andria, Miriam Gozalo, Björn Hagemeier, Daniel Mallmann, Josep Martrat, David Garciá Peréz, Angela Rumpl, Wolfgang Ziegler, and Csilla Zsigri. elasticLM: A Novel Approach for Software Licensing in Distributed Computing Infrastructures. *Proccedings of the IEEE CloudCom conference, Indianapolis, USA*, pages 67–74, 2010.

[30] Claudio Cacciari, Francesco D'andria, Bj orn Hagemeier, Daniel Mallmann, Josep Martrat, David Garciá Peréz, Angela Rumpl, Wolfgang Ziegler, and Csilla Zsigri. Software licenses as mobile objects in distributed computing environments. In *Proceedings of the CoreGRID Workshop, Euro-Par 2010*, pages 279–286, 2010.

[31] Claudio Cacciari, Daniel Mallmann, Csilla Zsigri, Francesco D'Andria, Björn Hagemeier, Angela Rumpl, Wolfgang Ziegler, and Josep Martrat. Sla-based management of software licenses as web service resources in distributed computing infrastructures. *Future Generation Computer Systems*, 28(8):1340 –

1349, 2012. Including Special sections SS: Trusting Software Behavior and SS: Economics of Computing Services.

[32] Roberto G Cascella, Lorenzo Blasi, Yvon Jegou, Massimo Coppola, and Christine Morin. Contrail: Distributed Application Deployment under SLA in Federated Heterogeneous Clouds. In *The Future Internet*, pages 91–103. Springer Berlin Heidelberg, Berlin, Heidelberg, May 2013.

[33] Jen-Hsiang Chen, Fahmida Abedin, Kuo-Ming Chao, Nick Godwin, Yinsheng Li, and Chen-Fang Tsai. A hybrid model for cloud providers and consumers to agree on QoS of cloud services. *Future Gener. Comput. Syst.*, 50:38–48, 2015.

[34] Cloud Standards Customer Council. Practical Guide to Cloud Service Agreements Version 2.0. Technical report, April 2015. Online at http://www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-Cloud-Service-Agreements.pdf, visited 13 May 2016.

[35] The Maturity Level of your IaaS Service Level Agreement - executive summary. Website. Online at http://www.cloudquadrants.com/home/135-iaas-summary-free-research.html, visited 31 July 2016.

[36] CloudQuadrants web site. Website. Online at http://www.cloudquadrants.com/, visited 31 July 2016.

[37] FIPA Technical Committee Communication. FIPA Contract Net Interaction Protocol Specification, FIPASpecification SC00029H. Technical report, Foundation for Intelligent Physical Agents, 2002. Online at http://www.fipa.org/specs/fipa00030/SC00029H.html, visited 10 May 2016.

[38] Contrail - open computing infrastructures for elastic services; project homepage. Website. Online at http://contrail-project.eu/, visited 18 May 2016.

[39] K Cristiano, R Gruber, V Keller, P Kuonnen, S Maffioletti, N Nellari, M-C Sawley, M Spada, T-M Tran, P Wieder, O Wäldrich, and W Ziegler. Integration of ISS into the VIOLA Meta-scheduling Environment. In S Gorlatch and M Danelutto, editors, *Integrated Research in Grid Computing, Proceedings of the CoreGRID Integration Workshop, Pisa, November 28-30 2005, Springer CoreGRID series, Volume 4*, pages 203–214, 2007.

[40] C-SIG - SLA standardisation guidelines fourth final draft. Technical report, September 2014. Online at http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?action=display&doc_id=6138, visited 30 April 2016.

[41] Karl Czajkowski, Ian T Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *8th International Workshop, JSSPP Edinburgh, Scotland, UK, July , Revised Papers*, pages 153–183, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[42] D-Grid. Website. Online at http://www.d-grid-gmbh.de/index.php?id=1&L=1, visited 30 May 2016.

[43] Gerard Damm, Greg Bain, John Timms, Laurent Philippart, Quan Huynh-Thu, Ron Roman, David Milham, and Tina O'Sullivan. SLA Management Handbook. Technical Report GB 917, January 2011.

[44] Data protection regulation. pages 1–88, April 2016. Online at http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN, visited 21 May 2016.

[45] DGSI - D-Grid Scheduler Interoperability. Website. Online at http://dgsi.d-grid.de, visited 30 April 2016.

[46] Giuseppe Di Modica, Valerio Regalbuto, Orazio Tomarchio, and Lorenzo Vita. Dynamic re-negotiations of SLA in service composition scenarios. *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, pages 359–366, 2007.

[47] Distributed Management Task Force. Website. Online at http://www.dmtf.org, visited 18 May 2016.

[48] Amazon EC2 Service Level Agreement. Website, June 2013. Online at http://aws.amazon.com/ec2-sla/, visited 18 May 2016.

[49] Thomas Eickermann, Wolfgang Frings, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. Co-allocation of MPI Jobs with the VIOLA Grid MetaScheduling Framework. In *Proceedings of the German e-Science Conference 2007. Proceedings, May 2. - 4. 2007 Baden-Baden, Germany*, May 2007.

[50] Ana Juan Ferrer, Francisco Hernández, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, Raül Sirvent, Jordi Guitart, Rosa M Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K Nair, George Kousiouris, Kleopatra Konstanteli, Theodora A Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, Nikolaus Forgó, Tabassum Sharif, and Craig Sheridan. OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1):66–77, 2012.

[51] S Figuerola, N Ciulli, M Leenheer, Y Demchenko, W Ziegler, and A Binczewski. PHOSPHORUS: Single-step on-demand services across multi-domain networks for e-science. In *Network Architectures, Management, and Applications V, Wuhan, China*, page 67842X, November 2007.

[52] I Foster, J Geisler, B Nickless, W Smith, and S. Tuecke. Software infrastructure for the I-WAY high-performance distributed computing experiment. In *High Performance Distributed Computing, 1996., Proceedings of 5th IEEE International Symposium on*, pages 562–571, 1996.

[53] I Foster and C Kesselman. *The grid: blueprint for a new computing infrastructure*. Advanced computing. Computer systems design. Morgan Kaufmann Publishers, 1999.

[54] Casey K Fung, Patrick C K Hung, Richard C Linger, and Gwendolyn H Walton. Extending Business Process Execution Language for Web Services with Service Level Agreements Expressed in Computational Quality Attributes. In *38th Hawaii International Conference on System Sciences*, pages 1–10, December 2005.

[55] Georgina Gallizo, Roland Kuebert, Karsten Oberle, Kleopatra Konstanteli, and Andreas Menychtas. Service Level Agreements in Virtualised Service Platforms. In Miriam Cunningham and Paul Cunningham, editors, *eChallenges e-2013 Conference Proceedings*, pages 1–8, September 2009.

[56] GLUE Specification, Version 2.0. Website. Accessible on OGF's web-site: http://www.ogf.org/documents/GFD.147.pdf, visited 28 April 2016.

[57] Íñigo Goiri, Ferran Julià, J Oriol Fitó, Mario Macías, and Jordi Guitart. Supporting CPU-based guarantees in cloud SLAs via resource-level QoS metrics. *Future Gener. Comput. Syst.*, 28(8):1295–1302, 2012.

[58] Grid Resource Allocation Agreement Protocol Working Group. Website. online at https://redmine.ogf.org/projects/graap-wg/, visited 19 May 2016.

[59] Hans Graux, Jos Dumortier, Patricia Ypma, Jasmine Simpson, Peter McNally, and Marc de Vries. Standards terms and performance criteria in service level agreements for cloud computing services. Technical report, September 2015.

[60] Les Green. Service level negotiation in a heterogeneous telecommunication environment. In *Proceedings of the Conference on Computing, Communications and Control Technologies 2004 (CCCT'04), Austin, Texas, USA, August 14-17, 2004*. Austin, TX, USA, August 2004.

[61] Andrew S. Grimshaw, Wm. A. Wulf, and CORPORATE The Legion Team. The legion vision of a worldwide virtual computer. *Commun. ACM*, 40(1):39–45, January 1997.

[62] Ralf Gruber, Vincent Keller, Pierre Kuonen, Marie-Christine Sawley, Basile Schaeli, Ali Tolou, Marc Torruella, and Trach-Minh Tran. Towards an Intelligent Grid Scheduling System. In *Parallel Processing and Applied Mathematics*, pages 751–757. Springer Berlin Heidelberg, Berlin, Heidelberg, September 2005.

[63] Ralf Gruber, Vincent Keller, Oliver Wäldrich, Philipp Wieder, Wolfgang Ziegler, and Pierre Manneback. Integration of grid cost model into ISS/VIOLA meta-scheduler environment. In *Euro-Par'06: Proceedings of the CoreGRID 2006, UNICORE Summit 2006, Petascale Computational Biology and Bioinformatics conference on Parallel processing*, pages 215–224. Springer-Verlag, August 2006.

[64] Globus Toolkit - The Globus Project. Website. Online at http://www.globus.org/toolkit, visited 10.June 2016.

[65] Björn Hagemeier, Daniel Mallmann, and Wolfgang Ziegler. Securing software licenses in a location independent manner. In *Cracow Grid Workshop '09*, pages 112–119, 2010.

[66] S. Hudert, T. Eymann, H. Ludwig, and G. Wirtz. A negotiation protocol description language for automated service level agreement negotiations. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 162–169, July 2009.

[67] Sebastian Hudert. From service markets to service economies - an infrastructure for protocol-genric sla negotiations. In *Grids and Service-Oriented Architectures for Service level Agreements*, pages 119–130. Springer CoreGRID Series, Volume 13, 2010.

[68] Sebastian Hudert, Heiko Ludwig, and Guido Wirtz. A Negotiation Protocol Framework for WS-Agreement. In *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, pages 1–6, March 2007.

[69] ISO/IEC JTC1/SC38 WG 3. Information technology – Cloud computing – Service level agreement (SLA) framework – Part 1: Overview and concepts. ISO/IEC, November 2014.

[70] ISO/IEC JTC1/SC38 WG 3. Information Technology - Cloud Computing – Service Level Agreement (SLA) Framework — Part 2: Metrics. ISO/IEC, March 2016.

[71] ISO/IEC JTC 1/SC 38 Cloud Computing and Distributed Platforms. Website. Online at http://www.iso.org/iso/iso_technical_committee.html?commid=601355/, visited 30 May 2016.

[72] FIPA Technical Committee Communication. FIPA Iterated Contract Net Interaction Protocol Specification, FIPASpecification SC00030H. Technical report, Foundation for Intelligent Physical Agents, 2002. Online at http://www.fipa.org/specs/fipa00030/SC00030H.html, visited 10 May 2016.

[73] ITIL - IT Infrastructure Library. Website. Online at http://www.itil-officialsite.com, visited 30 May 2016.

[74] Y. Jegou, P. Harsh, R.G. Cascella, F. Dudouet, and C. Morin. Managing ovf applications under sla constraints on contrail virtual execution platform. In *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, pages 399–405, 2012.

[75] Job Submission Description Language (JSDL) Specification, Version 1.0. Website. Accessible on OGF's web-site: http://www.ogf.org/documents/GFD.136.pdf, visited 28 April 2016.

[76] K.T. Kearney, F. Torelli, and C. Kotsokalis. Sla *: An abstract syntax for service level agreements. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 217–224, 2010.

[77] Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[78] Vincent Keller, Hassan Rasheed, Oliver Wäldrich, Wolfgang Ziegler, Ralf Gruber, Marie-Christine Sawley, and Philipp Wieder. Models and internals of the IANOS resource broker. *Computer Science, Research + Development*, 23(3-4):259–266, 2009.

[79] Dimosthenis Kyriazis. Cloud Computing Service Level Agreements. Technical report, European Commission Directorate General Communications Networks, Content and Technology Unit E2 - Software and Services, Cloud, July 2013. Online at https://ec.europa.eu/digital-agenda/en/news/cloud-computing-service-level-agreements-exploitation-research-results, visited 15 June 2016.

[80] D D Lamanna, J Skene, and W Emmerich. SLAng: a language for defining service level agreements. In *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, 2003. FTDCS 2003.*, pages 100–106. IEEE, 2003.

[81] Andy Lawrence, Karim Djemame, Oliver Wäldrich, Wolfgang Ziegler, and Csilla Zsigri. Using service level agreements for optimising cloud infrastructure services. In *ServiceWave'10: Proceedings of the 2010 international conference on Towards a service-based internet*. Springer-Verlag, December 2010.

[82] Hui Li. Challenges in SLA translation. Technical report, SLA@SOI, December 2009. Online at http://sla-at-soi.eu/wp-content/uploads/2009/12/ChallengesInSLATranslation.pdf, visited 30 May 2016.

[83] Jiadao Li, Oliver Wäldrich, and Wolfgang Ziegler. Towards SLA-based software licenses. In *From Grids to Service and Pervasive Computing*, pages 139–152. Springer CoreGRID Series, Volume 10, 2008.

[84] Kuan Lu. Planning and optimization during the life-cycle of service level agreements for cloud computing. 2015.

[85] Kuan Lu, Thomas Röblitz, Peter Chronz, and Constantinos Kotsokalis. *SLA-Based Planning for Multi-Domain Infrastructure as a Service*, pages 243–257. Springer New York, New York, NY, 2012.

[86] Kuan Lu, Ramin Yahyapour, Philipp Wieder, Edwin Yaqub, Monir Abdullah, Bernd Schloer, and Constantinos Kotsokalis. Fault-tolerant service level agreement lifecycle management in clouds using actor system. *Future Generation Computer Systems*, 54:247 – 259, 2016.

[87] Heiko Ludwig, Alexander Keller, Asit Dan, and Richard King. A Service Level Agreement Language for Dynamic Electronic Services. In *Proceedings of the Fourth IEEE International Workshop on Advanced Issues of ECommerce and WebBased Information Systems WECWIS 2002, Los Alamitos, CA, USA*, page 25, 2002.

[88] Heiko Ludwig, Toshiyuki Nakata, Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. Reliable Orchestration of Resources using WS-Agreement. In *International Conference on High Performance Computing and Communications (HPCC) 2006, München, Springer LNCS*, pages 753–762, September 2006.

[89] Adil Maarouf, Mahmoud El Hamlaoui, Abderrahim Marzouk, and Abdelkrim Haqiq. MDE Approach for the Establishment of a Service Level Agreements Monitoring by Trusted Third Party in the Cloud Computing. *International Journal of Grid and High Performance Computing (IJGHPC)*, 7(4):1–20, January 1.

[90] J. MacLaren, R. Sakellariou, K.T. Krishnakumar, J. Garibaldi, and D. Ouelhadj. Towards Service Level Agreement Based Scheduling on the Grid. Workshop on Planning and Scheduling for Web and Grid Services (in conjunction with ICAPS-04), 2004.

[91] P. Masche, P. Mckee, and B. Mitchell. The increasing role of service level agreements in B2B systems. In *2nd international conference on web information systems and technologies*. Citeseer, 2006.

[92] Peter Mell and Timthy Grance. The NIST Definition of Cloud Computing. Technical Report 800-145, September 2011. Online at http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, visited 14 May 2016.

[93] Zeqian Meng and John Brooke. Negotiation Protocol for Agile and Reliable E-science Collaboration. *2015 IEEE 11th International Conference on e-Science (e-Science)*, pages 292–295, November 2015.

[94] METEOR-S: Semantic Web Services and Processes. Website. Online at http://lsdis.cs.uga.edu/projects/meteor-s/, visited 30 May 2016.

[95] Andras Micsik, Henar Munoz Frutos, Ioannis Kotsiopoulos, and Bastian Koller. Semantically supported SLA negotiation. In *2009 10th IEEE/ACM International Conference on Grid Computing (GRID)*, pages 169–170. IEEE, August 2009.

[96] Carlos Müller, Ocatvio Martin-Diaz, Antonio Ruiz-Cortes, Resinas Manuel, and Pablo Fernandez. Improving Temporal-Awareness of WS-Agreement. In *Service-Oriented Computing–ICSOC 2007*, pages 193–206. ICSOC conference, Vienna, 2007, 2007.

[97] S. Nepal, J. Zic, and S. Chen. Wsla+: Web service level agreement language for collaborations. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, volume 2, pages 485–488, July 2008.

[98] NIST - National Institute of Standards and Technology. Website. Online at www.nist.gov, visited 10 June 2016.

[99] Cloud Computing Service Metrics Description. Technical Report Special Publication 500-307, National Institute of Standards and Technology (NIST), December 2014. Online at http://www.nist.gov/itl/cloud/upload/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf, visited 13 May 2016.

[100] OASIS - Organization for the Advancement of Structured Information Standards. Website. Online at http://www.oasis-open.org/, visited 18 July 2016.

[101] OCCI - Open Cloud Computing Interface. Website. Online at http://occi-wg.org/, visited 3 July 2016.

[102] Open Cloud Computing Interface – Service Level Agreements. Website. Online at https://redmine.ogf.org/attachments/224/slas.pdf, visited 20 May 2016.

[103] Open Grid Forum. Website. Online at http://www.ogf.org, visited 19 May 2016.

[104] OGSA - WSRF Basic Profile 1.0. Website. Online at https://http://www.ogf.org/documents/GFD.72.pdf, visited 30 May 2016.

[105] Nicole Oldham, Kunal Verma, Amit Sheth, and Farshad Hakimpour. Semantic WS-Agreement Partner Selection. In *Proceedings of the 15th international conference on World Wide Web - WWW '06*, page 697, New York, New York, USA, 2006. ACM Press.

[106] Cloud Computing Use Cases White Paper, July 2010.

[107] The Open Cloud Manifesto. Website. http://opencloudmanifesto.org/, visited 24 June 2016.

[108] OpenNebula - Open source data centre virtualisation. Website. Online at http://opennebula.org/start/, visited 7 June 2016.

[109] OPTIMIS - Optimised Infrastructure Services. Website. Online at http://www.optimis-project.eu, visited 20 May 2016.

[110] OPTIMIS - Project Webpages. Website. Online at http://optimis-project.eu/, visited 20 May 2016.

[111] Open Virtualization Format Specification Version 2.1.1. Website. Accessible on DMTF's web-site: http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.1.pdf, visited 18 May 2016.

[112] OVF - Open Virtualization Format. Website. Online at http://dmtf.org/standards/ovf, visited 21 May 2016.

[113] M Parkin, D. Kuo, and J. Brooke. A Framework Negotiation Protocol for Service Contracts. In *Services Computing, 2006. SCC '06. IEEE International Conference on*, pages 253–256, September 2006.

[114] Michael Parkin, Rosa M. Badia, Josep Martrat, Michael Parkin, Rosa M. Badia, and Josep Martrat. A comparison of sla use in six of the european commissions fp6 projects. Technical report, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, 2008.

[115] Michael Parkin, Peer Hasselmeyer, and Bastian Koller. An SLA Re-Negotiation Protocol. In *2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop NFPSLA-SOC'08*, November 2008.

[116] Pankesh Patel, Ajith Ranabahu, and Amit Sbeth. Service Level Agreements in Cloud Computing. In *OOPSLA 2009 - The International Conference on Object Oriented Programming, Systems, Languages and Application*, 2009.

[117] PHOSPHORUS - Lambda User Controlled Infrastructure for European Research. Website. Online at http://www.ist-phosphorus.eu/, visited 10 Juni 2016.

[118] Antoine Pichot, Philipp Wieder, Oliver Wäldrich, and Wolfgang Ziegler. Dynamic SLA Negotiation based on WS-Agreement. In *Proceedigs of the WEBIST conference, Funchal, Portugal, May 2008*, 2008.

[119] Antoine Pichot, Philipp Wieder, Oliver Wäldrich, and Wolfgang Ziegler. Towards dynamic service level agreement negotiation: an approach based on WS-Agreement. *Web information systems and technologies*, pages 107–119, 2009.

[120] Markus Pilz Pilz, Christoph Barz, Peter Martini, Christian de Waal, and Alexander Willner. ARGON: Reservation in Grid-enabled Networks. In . *DFN-Forum*, pages 1–11, March 2008. Online at https://www.dfn.de/fileadmin/3Beratung/DFN-Forum1/112-1130-1200.pdf, visited 31 May 2016.

[121] Tobias Pontz, Manfred Grauer, Roland Kuebert, Axel Tenschert, and Bastian Koller. Evaluation of Service Level Agrerment Approaches for Portfolio Management in the Financial Industry. *Proceedings of 10th IEEE / ACM International Conference on Grid Computing, Banff, Alberta, Canada, 2009*, pages 1–9, July 2009.

[122] Gerd Quecke and Wolfgang Ziegler. MeSch - An Approach to Resource Management in a Distributed Environment. In *Grid Computing — GRID 2000*, pages 47–54. Springer Berlin Heidelberg, Berlin, Heidelberg, December 2000.

[123] Thomas B. Quillinan, Kassidy P Clark, Martijn Warnier, Frances M.T. Brazier, and Omer Rana. Negotiation and Monitoring of Service Level Agreements. In Philipp Wieder, Ramin Yahyapour, and Wolfgang Ziegler, editors, *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 167–176. Springer US, 2010.

[124] Thomas B. Quillinan, Kassidy P. Clark, Martijn Warnier, and Frances M.T. Brazier. Negotiation and monitoring of service level agreements. In *Grids and Service-Oriented Architectures for Service level Agreements*, pages 141–150. Springer CoreGRID Series, Volume 13, 2010.

[125] Hassan Rasheed, Ralf Gruber, Vincent Keller, Pierre Kuonen, Philipp Wieder, Oliver Wäldrich, and Wolfgang Ziegler. IANOS: an intelligent application oriented scheduling framework for an HPCN Grid. In *Grid Computing: Achievements and Prospects*, pages 237–248. Springer CoreGRID Series, Volume 9, 2008.

[126] Hassan Rasheed, Angela Rumpl, Oliver Wäldrich, and Wolfgang Ziegler. A Standards-Based Approach for Negotiating Service QoS with Cloud Infrastructure Providers. In Paul Cunningham and Miriam Cunningham, editors, *eChallenges e-2012*, pages 1–9, September 2012.

[127] Hassan Rasheed, Angela Rumpl, Oliver Wäldrich, and Wolfgang Ziegler. Service Manifest: Scientific Report. Technical report, May 2012. Online at http://www.optimis-project.eu/sites/default/files/content-files/document/optimis-public-deliverable-service-manifest-scientific-report.pdf, visited 2 June 2016.

[128] Leonard Richardson and Sam Ruby. *RESTful Web Services. Web services for the real world*. O'Reilly Media, May 2007.

[129] Morris Riedel, Volker Sander, Philipp Wieder, and Jiulong Shan. Web Services Agreement based Resource Negotiation in UNICORE. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2005 Las Vegas, Nevada, USA - Volume 1, Workshop on Scheduling & Resource Management for Parallel & Distributed Systems - SRMPDS*, pages 31–37, 2005.

[130] Marcel Risch and Jörn Altmann. Enabling open cloud markets through ws-agreement extensions. In *Grids and Service-Oriented Architectures for Service level Agreements*, pages 105–117. Springer CoreGRID Series, Volume 13, 2010.

[131] Angela Rumpl, Oliver Wäldrich, and Wolfgang Ziegler. Extending WS-AGREEMENT with multi-round negotiation capability. In *Grids and Service-Oriented Architectures for Service level Agreements*, pages 89–103. Springer CoreGRID Series, Volume 13, 2010.

[132] K Saravanan and M Rajaram. An Exploratory Study of Cloud Service Level Agreements - State of the Art Review. *KSII Transactions on Internet and Information Systems*, 9(3):1–30, March 2015. Online at http://itiis.org/digital-library/manuscript/968/download, visited 30 April 2016.

[133] UNCITRAL Secretariat. Convention on Contracts for the International Sale of Goods. Website, 1980. online at http://www.uncitral.org/pdf/english/texts/sales/cisg/V1056997-CISG-e-book.pdf, visited 19 May 2016.

[134] Jan Seidel, Oliver Wäldrich, Philipp Wieder, Ramin Yahyapour, and Wolfgang Ziegler. Using SLA for Resource Management and Scheduling - A Survey. In *Grid Middleware and Services: Challenges and Solutions*, pages 334–347. Springer CoreGRID Series, Volume 8, 2008.

[135] W Shen, Hamada H Ghenniwa, and C Wang. Adaptive Negotiation for Agent-Based Grid Computing. In *AgentcitiesAAMAS*, pages 32–36. Bologna, Italy, 2002.

[136] Malcolm Sinton, Greg Bain, Debbie Burkett, Jane Hall, Peter Hucket, Ranveer Rathore, Tobey Trygar, and Lightsey Wallace. SLA Management Handbook - Service and Technology Example. Technical Report GB 917-3, TeleManagement Forum, January 2005.

[137] Dale Skeen. Nonblocking commit protocols. In *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, SIGMOD '81, pages 133–142, New York, NY, USA, 1981. ACM.

[138] SLA4D-Grid - Service Level Agreements for D-Grid. Website. Online at http://www.sla4d-grid.de, visited 31 May 2016.

[139] The SLAng project website. Website. Online at http://uclslang.sourceforge.net/, visited 31 May 2016.

[140] SLA-Ready project web site. Website. Online at http://www.sla-ready.eu/, visited 10 May 2016.

[141] SLALOM project web site. Website. Online at http://slalom-project.eu/, visited 10 May 2016.

[142] SLA@SOI project homepage. Website. Online at http://sla-at-soi.eu/, visited 28 May 2016.

[143] Reference Architecture for an SLA Management Framework. Whitepaper. Online at http://sla-at-soi.eu/wp-content/uploads/2010/12/SLA@SOI-Reference_Architecture.pdf/, visited 31 May 2016.

[144] SLA@SOI - SLA Model. Website. Online at http://wiki.sla-at-soi.eu/index.php?title=Workspace/Workpackages/Action_Line_A/A5_SLA_Management_%26_Foundations/TA5.1_SLA_Model, visited 10 May 2016.

[145] SmartLM - Grid-friendly software licensing for location independent application execution. Website. Online at http://www.smartlm.eu/, visited 10 June 2016.

[146] SPECS - Secure Provisioning of Cloud Services based on SLA Management. Website. Online at http://www.specs-project.eu/, visited 10 May 2016.

[147] Katerina Stamou, Jocelyn Aubert, Benjamin Gateau, and Jean-Henry Morin. Preliminary Requirements on Trusted Third Parties for Service Transactions in Cloud Environments. In *2013 46th Hawaii International Conference on System Sciences (HICSS)*, pages 4976–4983. IEEE, 2013.

[148] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and Ph. Wieder. Unicore — from project results to production grids. In Lucio Grandinetti, editor, *Grid Computing The New Frontier of High Performance Computing*, volume 14 of *Advances in Parallel Computing*, pages 357 – 376. North-Holland, 2005.

[149] Trusted and QoS-Aware Provision of Application Services. Website. Online at http://tapas.sourceforge.net/, visited 31 May 2016.

[150] TMF - TeleManagement Forum. Website. Online at http://www.tmforum.org/browse.aspxl, visited 18 July 2012.

[151] Johan Tordsson, Karim Djemame, Daniel Espling, Gregory Katsaros, Wolfgang Ziegler, Oliver Wäldrich, Kleopatra Konstanteli, Ali Sajjad, Muttukrishnan Rajarajan, Georgina Gallizo, and Srijith Nair. Towards holistic cloud management. In Jose Luis Vazquez-Poletti and Dana Pectu, editors, *European research activities in cloud computing*, pages 122–150. Cambridge Scholars Publishing, Newcastle, 2012.

[152] Topology and Orchestration Specification for Cloud Applications Version 1.0. Technical report, OASIS, March 2013. Online at http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf, visited 30 August 2016.

[153] Enabling End-to-End Cloud SLA Management. Technical Report TR178, TM Forum, October 2014. Online at https://www.tmforum.org/resources/technical-report-best-practice/tr178-enabling-end-to-end-cloud-sla-management-v2-0-2/#, visited 13 March 2016.

[154] UNICORE - Distributed computing and data resources. Website. Online at http://www.unicore.eu, visited 3 June 2016.

[155] USDL Pricing Vocabulary. Website. Online at https://github.com/linked-usdl/usdl-price/blob/master/usdl-price_v2.ttl, visited 30 May 2016.

[156] Linked USDL. Website. Online at http://linked-usdl.org/, visited 30 May 2016.

[157] VIOLA - Vertically Integrated Optical Testbed for Large Applications. Website. Online at https://www.dfn.de/projekte/gefoerderte-projekte/viola/, visited 10 Juni 2016.

[158] O Wäldrich, D Battré, F Brazier, K Clark, M Oey, A Papaspyrou, P Wieder, and W Ziegler. WS-Agreement Negotiation Version 1.0. Technical Report GFD.193, 2011.

[159] Oliver Wäldrich. *Orchestration of Resources in Distributed, Heterogeneous Grid Environments Using Dynamic Service Level Agreements*. PhD thesis, Technische Universität Dortmund, December 2011.

[160] Oliver Wäldrich, Philipp Wieder, and Wolfgang Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In *PPAM'05: Proceedings of the 6th international conference on Parallel Processing and Applied Mathematics*. Springer-Verlag, September 2005.

[161] Philipp Wieder, Oliver Wäldrich, Ramin Yahyapour, and Wolfgang Ziegler. Improving Workflow Execution through SLA-based Advance Reservation. In *Proceedings of the CoreGRID integration workshop, Krakow, October 19-20 2006, Springer CoreGRID Series, Volume 6*, pages 207–221, October 2006.

[162] Philipp Wieder, Wolfgang Ziegler, and Vincent Keller. IANOS - efficient use of HPC Grid resources. *ERCIM News*, (74):27–29, 2008.

[163] WSAG4J - WS-Agreement and WS-Agreement Negotiation Implementation for Java. Website. Online at http://wsag4j.sourceforge.net/site/index.html, visited 2 June 2016.

[164] Web Service Level Agreement (WSLA) Language Specification. Website. online at http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf, visited 19 May 2016.

[165] WSRF - Web Services Resource Framework. Website. Online at https://www.oasis-open.org/committees/wsrf/, visited 30 May 2016.

[166] Extensible Markup Language (XML) 1.0 (Fifth Edition). Website. Online at https://www.w3.org/TR/REC-xml, visited 15 June 2016.

[167] XSD - XML Schema Definition Language 1.1 Part 1: Structures. Website. Online at http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/, visited 30 May 2016.

[168] XSD - XML Schema Definition Language 1.1 Part 2: Datatypes. Website. Online at http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/, visited 30 May 2016.

[169] Jun Yan, Ryszard Kowalczyk, Jian Lin, Mohan B Chhetri, Suk Keong Goh, and Jianying Zhang. Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, 23(6):748–759, 2007.

[170] Sami Yangui, Iain-James Marshall, Jean-Pierre Laisne, and Samir Tata. CompatibleOne: The Open Source Cloud Broker. *Journal of Grid Computing*, 12(1):93–109, November 2013.

[171] E. Yaqub, R. Yahyapour, P. Wieder, C. Kotsokalis, K. Lu, and A. I. Jehangiri. Optimal negotiation of service level agreements for cloud-based services through autonomous agents. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 59–66, June 2014.

[172] Edwin Yaqub. Generic methods for adaptive management of service level agreements in cloud computing. 2015.

[173] Edwin Yaqub, Ramin Yahyapour, Philipp Wieder, and Kuan Lu. *A Protocol Development Framework for SLA Negotiations in Cloud and Service Computing*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[174] Wolfgang Ziegler. SLAs for Energy-Efficient Data Centres: The Standards-Based Approach of the OPTIMIS Project. In Jyrki Huusko, Hermann Meer, Sonja Klingert, and Andrey Somov, editors, *Energy Efficient Data Centers*, volume 7396 of *Lecture Notes in Computer Science*, pages 37–46. Springer Berlin Heidelberg, 2012.

[175] Wolfgang Ziegler, Philipp Wieder, and Dominic Battré. Extending WS-Agreement for dynamic negotiation of Service Level Agreements. Technical report, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, August 2008.

Paper I

A Proposal for WS-Agreement Negotiation[1]

Dominic Battré[*], Frances M.T. Brazier[‡], Kassidy P. Clark[‡], Michael Oey[‡], Alexander Papaspyrou[§], Oliver Wäldrich[†], Philipp Wieder[¶], Wolfgang Ziegler[†]

[*]Technische Universität Berlin, Complex and Distributed IT Systems
10587 Berlin, Germany
dominic.battre@tu-berlin.de

[†]Fraunhofer SCAI, Schloss Birlinghoven
53754 Sankt-Augustin, Germany
{oliver.waeldrich, wolfgang.ziegler}@scai.fraunhofer.de

[‡]Systems Engineering, Faculty of Technology, Policy and Management
Delft University of Technology, The Netherlands
{f.m.brazier, k.p.clark, m.a.oey}@tudelft.nl

[§]Robotics Research Institute, TU Dortmund University
Dortmund, Germany
alexander.papaspyrou@tu-dortmund.de

[¶]IT Management and Service Computing Group TU Dortmund University
Dortmund, Germany
philipp.wieder@udo.edu

Abstract: The Web Services Agreement specification defines a normative language to formulate Service Level Agreements and a basic protocol to expose service-level descriptions, validate service-level requests, and come to an agreement. This protocol, often called "take-it-or-leave-it", allows a service provider and a service consumer to decide whether to accept or reject a service offer. Although this approach is sufficient for a number of use cases, others exist with requirements for multi-step negotiation or the adaptation of an existing agreement.

In this paper, we describe the Web Services Agreement Negotiation protocol, a proposal by the Open Grid Forum to extend the existing specification. This proposal is the result of combining various research activities that have been conducted to define protocols for negotiating service levels or to supersede the existing "take-it-or-leave-it" protocol.

[1]By permission of IEEE

The main characteristics of this proposal are the multi-round negotiation capability, renegotiation capability, and compliance with the original specification.

Keywords: Negotiation Constraints, Open Grid Forum, Re- Negotiation, SLA Negotiation, WS-Agreement

A Proposal for WS-Agreement Negotiation

Dominic Battré*, Frances M.T. Brazier‡, Kassidy P. Clark‡, Michael Oey‡,
Alexander Papaspyrou§, Oliver Wäldrich†, Philipp Wieder¶, Wolfgang Ziegler†

*Technische Universität Berlin
Complex and Distributed IT Systems
10587 Berlin, Germany
dominic.battre@tu-berlin.de

†Fraunhofer SCAI
Schloss Birlinghoven
53754 Sankt-Augustin, Germany
{oliver.waeldrich, wolfgang.ziegler} @scai.fraunhofer.de

‡Systems Engineering
Faculty of Technology, Policy and Management
Delft University of Technology
The Netherlands
{f.m.brazier, k.p.clark, m.a.oey}@tudelft.nl

§Robotics Research Institute
TU Dortmund University
Dortmund, Germany
alexander.papaspyrou@tu-dortmund.de

¶IT Management and Service Computing Group
TU Dortmund University
Dortmund, Germany
philipp.wieder@udo.edu

***Abstract*—The Web Services Agreement specification defines a normative language to formulate Service Level Agreements and a basic protocol to expose service-level descriptions, validate service-level requests, and come to an agreement. This protocol, often called "take-it-or-leave-it", allows a service provider and a service consumer to decide whether to accept or reject a service offer. Although this approach is sufficient for a number of use cases, others exist with requirements for multi-step negotiation or the adaptation of an existing agreement.**

In this paper, we describe the Web Services Agreement Negotiation protocol, a proposal by the Open Grid Forum to extend the existing specification. This proposal is the result of combining various research activities that have been conducted to define protocols for negotiating service levels or to supersede the existing "take-it-or-leave-it" protocol. The main characteristics of this proposal are the multi-round negotiation capability, re-negotiation capability, and compliance with the original specification.

***Keywords*-Negotiation Constraints, Open Grid Forum, Re-Negotiation, SLA Negotiation, WS-Agreement**

I. Introduction

Service Level Agreements (SLAs) are one central pillar of Service Level Management solutions. In this area, industry best practices and standards like the TeleManagement Forum's SLA Handbook [1] or the IT Infrastructure Library [2] exist for quite some time and are adopted by various stakeholders. They cover the full life-cycle of Service Level Agreements, including steps like the agreement on Quality-of-Service terms and the resulting provision of the respective services.

One thing missing was the possibility to automatically negotiate SLAs that adhere to a normative format. The Web Services Agreement Specification (WS-Agreement [3]) provides one solution to close that gap by defining a normative language to formulate Service Level Agreements and a basic protocol to come to an agreement. With WS-Agreement being adopted by a diversity of stakeholders [4], it became evident that the basic negotiation protocol, as described in the related work section, does not fulfill the requirements of all stakeholder and usage scenarios since it realises a single shot process ("take-it-or-leave-it") without any negotiation. Therefore, the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG), the home of WS-Agreement at the Open Grid Forum (OGF[1]), gathered these requirements and came up

[1] http://www.ogf.org/

with a proposal for a SLA negotiation protocol called WS-Agreement Negotiation. In this paper, we describe the final draft version of this protocol.

The remainder of this paper is organized in the following way: We introduce work related to SLA negotiation in the next section and then describe the advance reservation use case, which gives one motivating example for the specification of WS-Agreement Negotiation. In Section IV, we show the negotiation model that is the basis of the actual protocol, which we describe in detail in Section V. Following this, we provide a number of examples on how to apply the protocol (Section VI) and then conclude with an outlook sketching the next steps necessary to make this proposal an OGF recommendation.

II. Related Work

Negotiation is a widely studied topic and there are numerous publications addressing different aspects, e.g. [5] is a general purpose negotiation journal, [6] is a survey about negotiation in distributed resource management systems, while [7] and [8] discuss aspects of service negotiation in the Grid. In our context and in the simplest case, a user's job has to be executed and the Grid scheduler has to select between different target systems. If all systems are identical and only one parameter influences the selection, i.e. price, this case is similar to a typical business negotiation between one buyer and several sellers. An auctioning mechanism like the ones described in [9] can be used. Of course, we take the point of view of an end user, if we look at things from a resource provider's point of view, we have several jobs that compete for one resource, i.e. several buyers and one seller. If we look at the scheduler's point of view, we have many jobs that compete for several resources, i.e. many buyers and many sellers. Buyya [9] (page 36) also surveyed several distributed resource management systems based on price.

Another focus of research during the last years was on automatic negotiation of SLAs, which is a complex and time consuming process [10]–[12], when even two users have to find an agreement on multiple criteria. Imagine how difficult the problem becomes when multiple entities have to reach an agreement as presented one of the oldest approaches for SLA negotiation [13]. When at least two resources are needed at the same time to run a job, e.g. a network connection and a processing resource, several steps have to be performed before reaching an agreement between the resource providers and the consumer. However, even negotiating QoS and availability for a single resource can require several steps to reach an agreement. Green [12] cites mainly two frameworks for automatic negotiation: ontologies and web services. According to him automated negotiation has three main considerations: The negotiation protocol, the negotiation objects and the decision-making models. He considers two options existing in order to achieve this type of negotiation. One option is for the originating agent to negotiate separately with each Autonomous System (AS) along each potential path to ensure that an end-to-end path is available. The dominant choice however, is to use a cascaded approach where each AS is responsible for the entire path downstream of itself. This approach enhances agent autonomy as it is only responsible for its immediate links. The autonomy of the cascaded approach struggles however with the issue of price. In a cascading scenario an intelligent agent would need to know the utility functions of all the downstream domains if the best price combination is to be determined, which is private information. In contrast, in this paper we limited the scope to protocols that permit the negotiation of agreements between each two parties based on WS-Agreement rather than tackling the full complexity of automated negotiation. These individual bilateral agreements might then be combined into one single agreement. The approach for negotiation of agreements presented in this paper is aiming to provide a generic protocol that can be used in different scenarios, e.g. bilateral or multilateral negotiations, agent-based negotiations or auction-style negotiations.

A negotiation protocol suitable for service composition has been proposed by Jun Yan et al. [14]. Service Level Agreements for a service composition are established through autonomous agent negotiation. A framework is proposed in which the service consumer is represented by a set of agents who negotiate quality of service constraints with the service providers for various services in the composition. Based on this framework, the authors propose a new negotiation protocol to support coordinated negotiation. However, this approach does not consider the current WS-Agreement specification.

More recent research on SLA negotiation based on WS-Agreement is addressing unreliability of message transmission in distributed environments. Parkin et al propose a protocol based on the principles of contract law and capable to cope with the imperfect message transmission layer [15]. In [16] the authors present early research on extending WS-Agreement with negotiation capabilities taking into account the symmetrical approach WS-Agreement assumes for the roles of agreement initiator and agreement provider.

III. The Advance Reservation of Compute Resources Use Case

In the scenario we select as an example, a service provider offering computing resources to customers. The computing resource service consists of a job submission service and portal application to manage the job submission service. The job submission service is a web service that provides methods for submitting and managing computing jobs, which are exposed via Web Service Description Language (WSDL) port types. The portal application provides methods to manage the job submission service, including user-related and resource-related operations, like updating a user profile or getting information about the resource consumption respectively.

Between the service provider and the consumers agreements are negotiated to have a shared understanding of Quality-of-Service provided and the obligations of either party. In the specific case here, it is of utmost importance to the consumer that the service provisioning starts at a certain point in time and lasts for a well-defined duration. To achieve this, the service provider offers the capability to reserve computing resources in

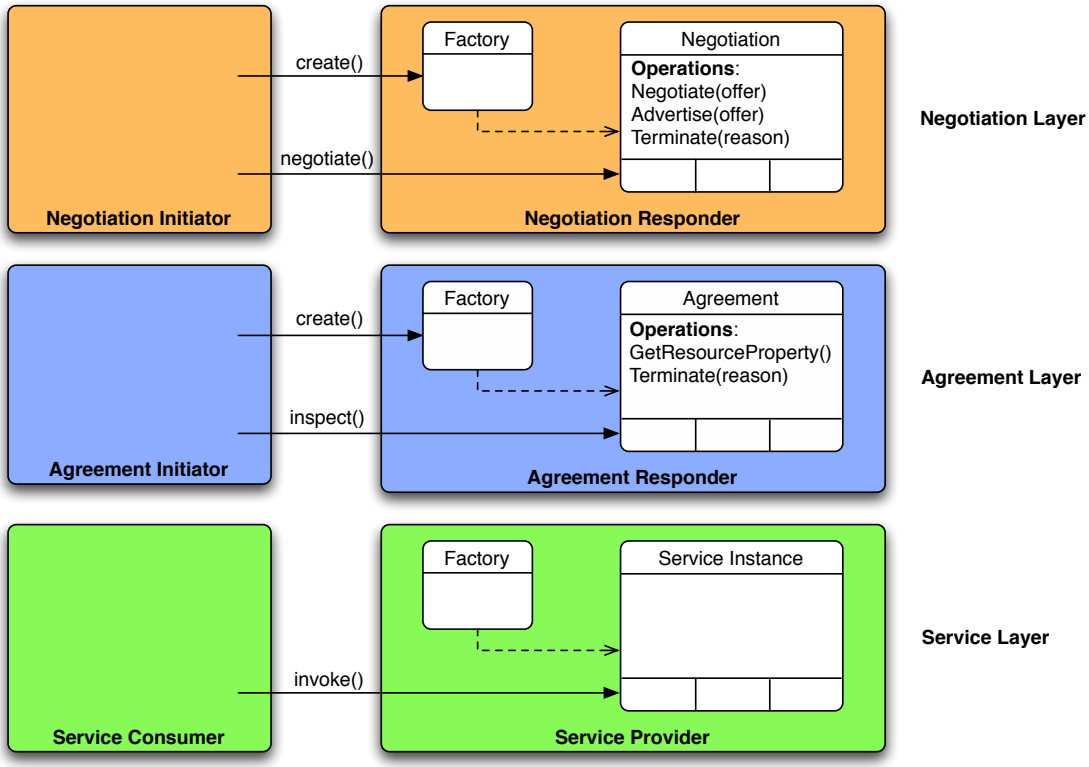


Fig. 1. Conceptual overview of the layered negotiation model

advance. In the case of the example the advance reservation is carried out through negotiation and governed by an agreement. The resource provisioning model is implementation-specific; whether resources are exclusively dedicated to a user, prediction models or preemption are used is up to the resource provider. Technically, the computing resource provider offers available computing resources via an agreement template [3]. The template includes the description of the service and its guarantees, as well as a set of options the customer can choose from. The service description contains the available computing resources and the time frame the resources are available (and hence can be reserved in advance). Moreover, the resource provider may offer different service levels for the computing resource service letting the customer e.g. choose between different classes of service availability (common values are specified as a the range between x% and y%) or average service response time (e.g. specified using a service-class name like gold, silver and bronze). The Quality-of-Service parameters can be specified separately for the job submission service and the portal application. The price of the service offer is then dependent on type and number of the selected computing resources and the respective service levels.

Potentially, the template provides many possibilities to parameterize the computing resource service. For example, the template contains parameters, such as pricing, that are dependent on the resources chosen and the service quality guaranteed. Once a customer has filled in all requirements into the SLA template, he sends the offer to the resource provider. The provider then checks whether the requested service can be provisioned at the requested time (i.e. whether the reservation of the resources can be carried out in advance). In case the service can be provided it sends back a completed counter-offer with the pricing information to the customer that in turn can now choose to create a negotiated agreement based on the offer. In case the resource provider is not able to fulfil all the requirements stated by the customer, it can also send back a counter-offer indicating a service quality it is able to provide instead. A customer has requested, for example, 128 nodes with 8GB memory in a given time frame, but the resource provider could not fulfil this request at this time. Instead the provider sends back a counter-offers for 96 nodes with 8GB memory and 32 nodes with 6GB memory for a lower price. The customer then may now choose to accept the counter-offer or any part of it purchasing the remaining resources somewhere else. The process of filling in all required fields of a negotiation offer can take multiple rounds.

Fig. 2. The Structure of a Negotiation Offer

At a later point in time (with an agreed upon SLA governing the job execution), the customer may recognise that he requires more resources to complete its computation. In that case the customer may initiate a re-negotiation of the agreement.

As there is currently no negotiation protocol that is compliant to WS-Agreement, offering multi-round negotiation and providing re-negotiation mechanisms, the GRAAP-WG specifies the negotiation protocol at hand.

IV. The Service Level Agreement Negotiation Model

As described in Section II a number of different approaches and models for negotiation have been presented over the last couple of years. The GRAAP working group of the Open Grid Forum spent quite some time in evaluating the use of different approaches for the negotiation of SLAs. Moreover, the group also defined and evaluated a number of models since 2008. Finally, - after the decision to provide negotiation on top of WS-Agreement rather than modifying WS-Agreement to include negotiation - the negotiation model presented in this section was developed and refined during several working group meetings at the Open Grid Forum and several dedicated small workshops. Major requirements for the development process were compliance with WS-Agreement, support for bilateral and multilateral negotiations, domain independency allowing to use the protocol in different environments, e.g. auctions or agent-based infrastructures, the possibility to later easily exchange the protocol against another one if more (specific) negotiation protocols evolve. Finally, the protocol should be symmetric with respect to the parties negotiating, service provider and service consumer. Details of the discussion process and how the protocol evolved can be found in the working group's Gridforge Project[2].

The WS-Agreement negotiation model consists of three layers, the negotiation layer, the agreement layer and the service layer. These layers are depicted in Figure 1.

There is a clear separation between these three layers in the negotiation model. The negotiation layer sits on top of the agreement layer. Therefore, it is decoupled from the agreement layer and the service layer. By that, the negotiation layer may change independently of the agreement layer and be replaced by another negotiation layer that may be better suited for specific negotiation scenarios. Agreement layer and service layer are modelled according to the WS-Agreement specification [3].

A. Negotiation Layer

The negotiation layer provides a protocol and a language to negotiate agreement offers and counter-offers, and to create agreements based on negotiated offers. The negotiation process comprises the exchange of offer and counter-offers. Negotiation offers, as defined in Section V-C, are non-binding by nature. Therefore, negotiated offers do not make any promises that a subsequent agreement based on a negotiated offer will be created. They only indicate the willingness of two negotiating parties to accept a subsequent creation of an agreement. However, WS-Agreement Negotiation can be extended to realize binding negotiation processes, but this is not in the focus of the current work and specification.

Once no further (counter-)offers are requested by one of the parties, negotiated agreements are then created by calling either the `createAgreement` or the `createPendingAgreement` operation on the agreement layer's Agreement Responder Factory instance (see [3] for details on WS-Agreement's operations).

B. Agreement Layer

The agreement layer provides the basic functionality to create and monitor agreements. It provides a protocol and a language defined in the WS-Agreement specification, to which we refer for further details.

C. Service Layer

At the service layer the actual service defined by an agreement is provided. This service may or may not be a web service. Moreover, a service defined by an agreement may consist of multiple services, e.g. a service for resource provisioning may consist of the provisioning service and a monitoring service for the provided resources. The service execution on the service layer is governed by the agreement layer.

V. WS-Agreement Negotiation

A negotiation (as depicted in Figure 1) is a service instance that is used by two negotiating parties to exchange information in order to come to a common understanding of valid agreement offers. In the process of a negotiation the two

[2]https://forge.gridforum.org/sf/projects/graap-wg

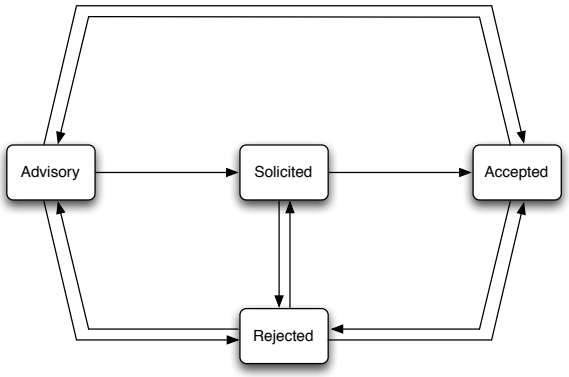


Fig. 3. Negotiation State Machine

negotiating parties exchange negotiation offers and indicate their goals and requirements. A negotiation may be limited in lifetime or rounds of negotiation. These limitations are defined in the negotiation context as described in Section V-B. In the following sections the key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' are to be interpreted as described in RFC 2119 [17].

A. Protocol

The protocol is providing support for bilateral negotiation of agreement offers. It is designed as a symmetric protocol, thus maintaining the inherent symmetry of the basic WS-Agreement protocol. In fact, it was an explicit design goal to support full symmetry regarding rights, obligations and capabilities of service consumer and service provider.

B. Negotiation Context

The negotiation context defines the roles of the negotiation participants, their obligations, and the nature of the negotiation process. Since a negotiation is a bi-lateral process, the roles of each participating party must be clearly defined. In general a negotiation process can either refer to the negotiation of new agreements or the re-negotiation of an existing agreement. Therefore, the type of the negotiation must be defined in the negotiation context. Moreover, the negotiation context defines the roles of the parties participating in the negotiation process. The negotiation participants must acknowledge these parameters for the entire negotiation process. The following listing shows a pseudo-schema[3] of the elements included in the negotiation context:

```
<wsag-neg:NegotiationContext>
    <wsag-neg:NegotiationType>
        wsag-neg:NegotiationType
    </wsag-neg:NegotiationType>
    <wsag-neg:ExpirationTime>
        xsd:dateTime
    </wsag-neg:ExpirationTime> ?
    <wsag-neg:NegotiationInitiator>
    xsd:anyType
    </wsag-neg:NegotiationInitiator> ?
    <wsag-neg:NegotiationResponder>
          xsd:anyType
    </wsag-neg:NegotiationResponder> ?
    <wsag-neg:AgreementResponder>
          wsag-neg:NegotiationRoleType
    </wsag-neg:AgreementResponder>
    <wsag-neg:AgreementFactoryEPR>
          wsa:EndpointReferenceType
    </wsag-neg:AgreementFactoryEPR> ?
</wsag-neg:NegotiationContext>
```

[3]The pseudo-schema uses BNF-style conventions for attributes and elements: '?' denotes zero or one occurrences, '*' denotes zero or more occurrences, and '+' denotes one or more occurrences.

The *NegotiationType* specifies the nature of the negotiation process, i.e. either the negotiation of a new agreement or the re-negotiation of an existing agreement. In the latter case, a reference to the agreement that is re-negotiated is provided.

Independent of the type of negotiation, every negotiation instance may carry an OPTIONAL *ExpirationTime*, which specifies the lifetime of the negotiation instance. If specified, the negotiation instance is accessible until the given time. After its lifetime, the negotiation instance is no longer accessible. For the *ExpirationTime* it is beneficial for the consistent interpretation of the *ExpirationTime* by the different parties if their clocks are synchronised as today usually realised with the Network Time Protocol (NTP).

NegotiationInitiator This OPTIONAL element identifies the initiator of the negotiation process. The negotiation initiator element can be an URI or an Endpoint Reference that can be used to contact the initiator. It can also be a distinguished name identifying the initiator in a security context.

NegotiationResponder This OPTIONAL element identifies the party that responds to the initiation of the negotiation process. This party implements the NegotiationFactory port type of this specification. This element can be an URI or an Endpoint Reference that can be used to contact the negotiation responder. It can also be a distinguished name identifying the negotiation responder in a security context.

AgreementResponder This REQUIRED element identifies the party in the negotiation process that acts on behalf of the agreement responder. This element can either take the value NegotiationInitiator or NegotiationResponder. The default value is NegotiationResponder, The party identified as agreement responder MUST provide a reference to the AgreementFactory (PendingAgreementFactory) it acts on the behalf of in the negotiation context by using the AgreementFactoryEPR element.

AgreementFactoryEPR This OPTIONAL element identifies the endpoint reference of the agreement factory that SHOULD be used to create agreements based on the negotiated agreement offers. After an agreement offer was successfully nego-

tiated, the party identified as agreement initiator MAY create a new agreement with the referenced factory.

A negotiation process comprises the exchange of offers and counter-offers. Counter-offers are created based on existing offers. An initial offer is created on the basis of an Agreement Template. The structure of a negotiation offer is basically the same as the structure of an Agreement Template. Agreement templates are defined in the Agreement Template and Creation Constraints section of the WS-Agreement specification. However, a negotiation offer contains the additional elements Negotiation Offer Context and Negotiation Constraints.

C. Structure of an Offer

In order to negotiate the content of an agreement, negotiation offers are exchanged between an agreement initiator and an agreement responder. In case one of the negotiating parties receives a negotiation offer, this party evaluates the offer and creates zero or more counter-offers, which are then sent back to the negotiation participant. The basic structure of a negotiation offer is shown in Figure 2.

A negotiation offer has basically the same structure as an agreement template [3], but in addition, the negotiation offer contains two more elements: a Negotiation Offer Id and a Negotiation Context (see Section V-B).

A Negotiation Offer also contains a Negotiation Constraints section. The Negotiation Constraints define the structure, valid ranges or distinct values that Service Terms may take in a counter-offer. The Negotiation Constraints of a negotiation offer must hold true for every counter-offer. In a negotiation process, however, the Creation Constraints MAY change during the advance of the negotiation. For example, if the negotiation initiator chooses one specific Service Term out of a set of Service Terms (ExactlyOne), a negotiation responder may adopt to this choice by changing the Creation Constrains section in a Counter Offer.

Negotiation Constraints are structurally identical to the Creation Constraints defined in an agreement template. Creation Constraints are defined in the section Agreement Template and Creation Constraints of the WS-Agreement specification.

D. Negotiation Offer States

During the negotiation process the content of Agreement Offers is negotiated before an agreement is created. A negotiated agreement is created by the party identified as Agreement Initiator in the negotiation context. A valid negotiated agreement offer MUST have the state Agreed when a new negotiated agreement is created. Figure 3 illustrates the states that negotiation offers can have and the valid state transitions.

1) Advisory State: The Advisory State identifies negotiation offers with no further obligations associated with. Offers in the Advisory State usually contain elements that are currently not specified. Therefore, these offers require further negotiation.

2) Solicited State: The Solicited State bears no obligations for an offer, but it requires that counter offers are either in the Accepted or the Rejected State. Solicited offers indicate that a negotiation participant wants to converge the negotiation process and requests only counter offers that can be accepted as is, e.g. where no further negotiation of the counter offers is required.

3) Accepted State: The Accepted State indicates that a negotiation participant accepts a negotiation offer as is. All details of a negotiation offer are specified and no further negotiation is required. However, since the negotiated offers are non-binding, there is no guarantee that a subsequent agreement is created. Augmented negotiation protocols may be created based on this specification to address binding negotiations.

4) Rejected State: If a negotiation offer is rejected, it is sent back to the inquiring party with the rejected state. The negotiation offer MAY contain a domain specific reason why it was rejected. Negotiation offers that are marked as rejected MUST NOT be used to create an agreement. However, they MAY be used to continue the negotiation process by taking into account the reason for rejecting the offer.

VI. Different Negotiation Examples

In this section a detailed description of the Negotiation Factory and the Negotiation port types is given through a set of example scenarios. These port types can be used in different combinations in order to support a wide range of signalling scenarios. The presented signalling scenarios are not meant to cover all possible combinations of the port types. They are presented here to illustrate possible negotiation scenarios and how these scenarios are mapped to specific deployments of WS-Agreement Negotiation. Furthermore, the interaction of the negotiation layer and the agreement layer is discussed.

A. Basic Client–Server Example

The simple client-server negotiation is an asymmetric signalling scenario, where a server implements the Negotiation Factory and Negotiation port types. The negotiation process itself is driven by the client. The client initiates a negotiation by calling the server's `initiateNegotiation` operation of the Negotiation Factory. After a new negotiation is created, the client queries the available templates from the Negotiation Responder that serve as initial templates for a negotiation offer. It uses these templates to create new negotiation offers and sends these offers to the server via the negotiate method of the Negotiation port type. The server may create one or more counter-offers for each offer that is part of the negotiate call. The server itself has a passive role in this negotiation process since it cannot actively influence the course of a negotiation, i.e. it can only react to negotiation requests. The process of negotiation is depicted in Figure 4.

B. Bilateral Negotiation with Asymmetric Agreement Layer

In a bilateral negotiation both parties can actively participate in the negotiation process. Both parties implement the WS-Agreement Negotiation port types. The process of initiating a bilateral negotiation is as follows. The Negotiation Initiator creates a new negotiation instance that implements the WS-Agreement Negotiation port type. It then sends an `initiateNegotiation` request

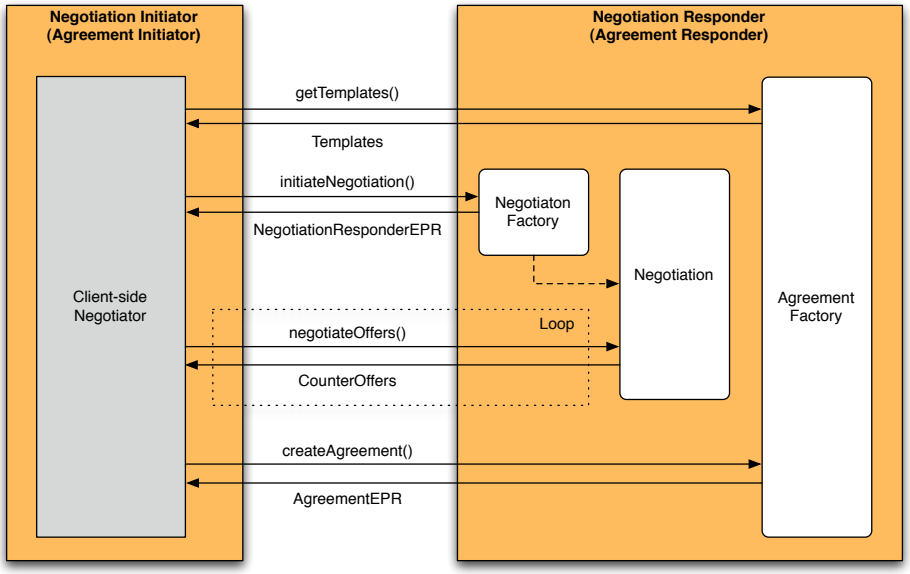


Fig. 4. Basic Client–Server Example showing Negotiation Layer and Agreement Factory (Asymmetric Deployment of the WS-Negotiation Port Types)

to the NegotiationFactory of the Negotiation Responder. The `initiateNegotiation` request includes an endpoint reference to the negotiation instance that was created beforehand. Moreover, it contains the negotiation context that defines the roles of each party in a negotiation, e.g. which party is the initiator and which is the responder of the agreements that are negotiated. In a bilateral negotiation process, the agreement templates that are used to create offers are provided by the agreement factory referenced in the negotiation context. The agreement initiator should query the available agreement templates from the agreement factory in order to create negotiation offers based on the provided templates. After a new Negotiation instance was created, the context of a negotiation must not change. Both parties participating in a negotiation process may actively send negotiation requests to the other party. It is not required that the initiator of a negotiation is also the initiator of the subsequent agreement. These roles may vary in different negotiation scenarios.

In the negotiation scenario depicted in Figure 5 the negotiation initiator is also the initiator of the subsequent agreements. It starts a negotiation process by retrieving the templates provided by the responder. Then the initiator notifies the responder of the offers it is willing to negotiate by calling the responder's `advertise` method. Now the negotiation responder takes an active role in the negotiation process by sending offers to the initiator. After several rounds of negotiation the initiator may decide to create an agreement based on one of the negotiated offer. It therefore calls the `createAgreement` method of the negotiation responder. The input of the `createAgreement` operation includes a critical extension that is the context of the negotiated offer that the initiator uses to create an agreement.

C. Re-Negotiation of Agreements with Symmetric Agreement Layer

In general the re-negotiation of an existing agreement follows the same signalling pattern as the negotiation of an agreement. If an existing agreement is re-negotiated, the initiator of the original agreement should match the initiator of the re-negotiated agreement, so that the roles and obligations match the original agreement (see Figure 6). The roles and the responsibilities of the negotiating parties are defined in the negotiation context, when a new negotiation is created. The negotiation context also includes an endpoint reference to the existing responder agreement. In a symmetric signalling scenario, the negotiation context may additionally include a reference to the original initiator agreement. After a new re-negotiation process has been initiated, both parties start to negotiate the contents of the agreement offer that can be used to create a re-negotiated agreement. When they succeeded to negotiate a suitable offer, the initiator of the negotiated agreement creates a new agreement by invoking the `createAgreement` (`createPendingAgreement`) method of the responders Agreement Factory (Pending Agreement Factory) instance. When a re-negotiated agreement is created, the original agreement must transition into the COMPLETED state.

The layout of the agreement layer may either be asymmetric or symmetric. In case of a symmetric layout of the agreement layer, the re-negotiated agreement

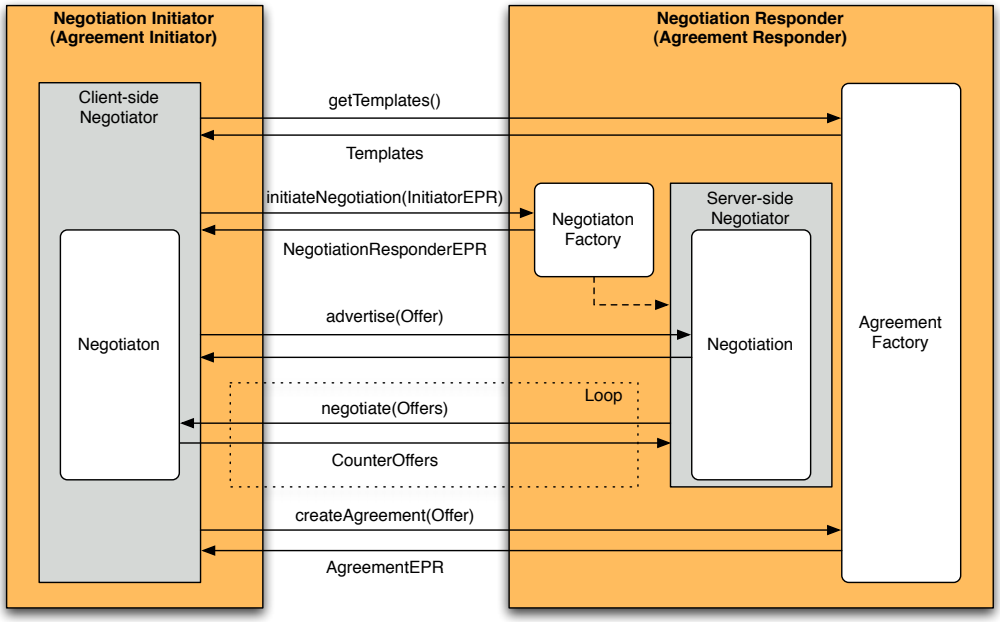


Fig. 5. Bilateral negotiation (Symmetric deployment of WS-Agreement Negotiation, where the Negotiation Initiator is also the Agreement Initiator and the Negotiation Responder is the Agreement Responder. Both Parties have an Active Role in the Negotiation Process.)

initiator creates an instance of the re-negotiated agreement before the `createAgreement` method (`createPendingAgreement` method) of the responder's agreement factory instance is invoked. This agreement must be in PENDING state until the responder has either accepted or rejected the creation of the re-negotiated agreement. After the initiator received the agreement responder's decision, the state of the pending agreement is updated accordingly. When a re-negotiated agreement is accepted, both parties must update the state of their original agreement instance to COMPLETED. Differences in the state of the original and re-negotiated agreements are handled in domain specific manner, e.g. by applying state replication, different levels of escalation or dispute handling.

VII. Discussion and Outlook

We described a negotiation protocol for Service Level Agreements that offers multi-round negotiation and re-negotiation capabilities. Furthermore, it is compliant with the WS-Agreement proposed recommendation and does not require modifications of the WS-Agreement specification v1.0. The protocol proposed has been under discussion in the GRAAP working group of the Open Grid Forum and is now ready to be submitted to the public comment process of the Open Grid Forum. The authors expect the current draft including eventual modifications resulting from the public comment period to become a proposed recommendation of the Open Grid Forum by end of 2010.

The work for defining the next version of WS-Agreement decoupling of the basic protocol from the language for creating agreements along with minor improvements will start by end of 2010. This major change will render WS-Agreement more flexible allowing to negotiate and create agreements with domain specific protocols without sacrificing compatibility.

VIII. Acknowledgements

Some of the work reported in this paper has been funded by the European Commissions ICT programme within the FP6 CoreGRID Network of Excellence under grant #004265 and in the FP7 project SmartLM under grant #216759. This paper also includes work funded by the German Federal Ministry of Education and Research through the D-Grid project under grant #01AK800A. Last not least the authors gracefully acknowledge the stimulating discussions in the GRAAP-WG.

References

[1] T. T. Forum, "SLA Management Handbook, Volume 2, Concepts and Principles," Heidelberg, Germany, 2005, release 2.5.

[2] R. Addy, *Effective IT Service Management – To ITIL and Beyond!* Morristown, New Jersey, United States: Springer-Verlag: Berlin, 2007.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," Open Grid Forum, Grid Forum Document GFD.107, 2007.

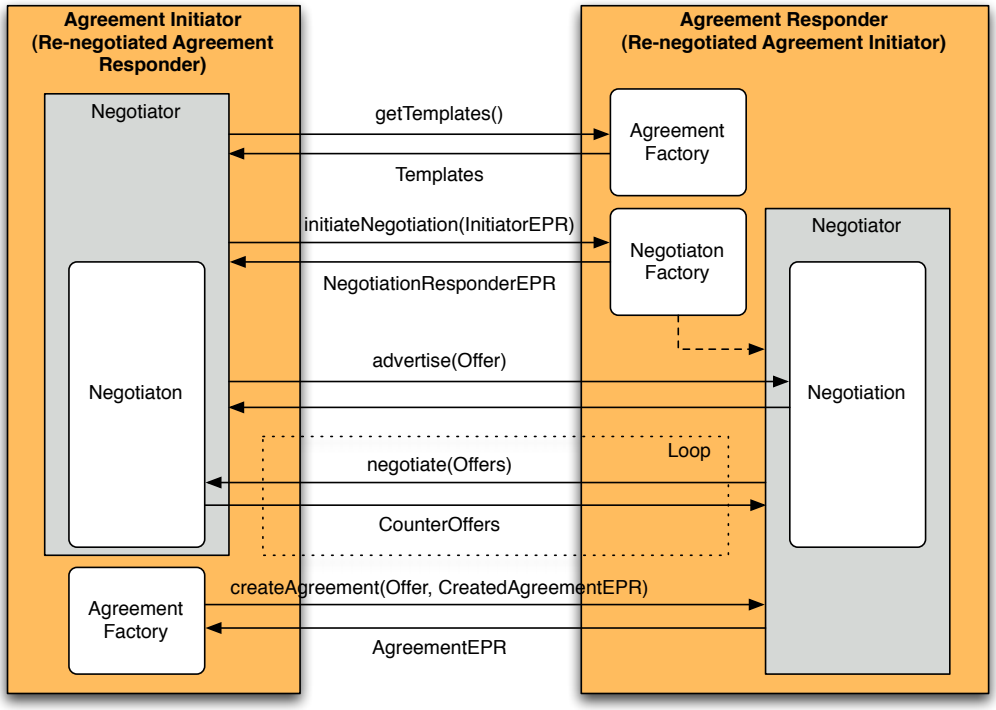


Fig. 6. Re-Negotiation of Agreements (Symmetric signalling on the Negotiation and Agreement Layer. Both parties implement the WS-Agreement Negotiation and WS-Agreement port types. Moreover, both parties have their own instance of the original agreement. After the negotiation process, the responder of the original agreement creates the re-negotiated agreement.)

[4] J. Seidel, O. Wäldrich, P. Wieder, R. Yahyapour, and W. Ziegler, "SLA for Resource Management and Scheduling - A Survey," in *Grid Middleware and Services: Challenges and Solutions (Proceedings of the Usage of Service Level Agreements in Grids Workshop in conjunction with the 8th IEEE International Conference on Grid Computing (Grid 2007))*, ser. CoreGrid series 8, D. Talia, R. Yahyapour, and W. Ziegler, Eds. Springer US, 2008.

[5] M. Shakun, Ed., *Group Decision and Negotiation*. Springer Netherlands, 2002.

[6] C. Briquet and P.-A. de Marneffe, "Grid resource negotiation: survey with a machine learning perspective," in *PDCN'06: Proceedings of the 24th IASTED international conference on Parallel and distributed computing and networks*. Anaheim, CA, USA: ACTA Press, 2006, pp. 17–22.

[7] D. Kuo, M. Parkin, and J. Brooke, "Negotiating Contracts on the Grid," in *Exploiting the Knowledge Economy - Issues, Applications, Case Studies, Volume 3, Proceedings of the eChallenges 2006 (e-2006) Conference*. Amsterdam, The Netherlands: IOS Press, 2006.

[8] ——, "A Framework & Negotiation Protocol for Service Contracts," in *Proceedings of the 2006 IEEE International Conference on Services Computing (SCC 2006)*, 2006, pp. 253–256.

[9] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis*. Melbourne, Australia: Monash University, 2002.

[10] N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge, "Automated Negotiation: Prospects, Methods and Challenges," *Group Decision and Negotiation*, vol. 10, no. 2, pp. 199–215, March 2001.

[11] W. Shen, H. H. Ghenniwa, and C. Wang, "Adaptive Negotiation for Agent-Based Grid Computing," in *Proceedings of AAMAS2002 workshop on agentcities: Challenges in Open Agent Environments*, Bologna, Italy, 2002, pp. 32–36.

[12] L. Green, "Service level negotiation in a heterogeneous telecommunication environment," in *Proceeding International Conference on Computing, Communications and Control Technologies (CCCT04)*, Austin, TX, USA, August 2004.

[13] K. Czajkowski, I. T. Foster, C. Kesselman, V. Sander, and S. Tuecke, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems," in *JSSPP '02: Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK: Springer-Verlag, 2002, pp. 153–183.

[14] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. K. Goh, and J. Zhang, "Autonomous service level agreement negotiation for service composition provision," *Future Gener. Comput. Syst.*, vol. 23, no. 6, pp. 748–759, 2007.

[15] M. Parkin, P. Hasselmeyer, and B. Koller, "An SLA Re-Negotiation Protocol," 2008, proceedings of the 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop NFPSLA-SOC'08.

[16] W. Ziegler, P. Wieder, and D. Battré, "ExtendingWS-Agreement for dynamic negotiation of Service Level Agreements," Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, Tech. Rep. TR-0172, August 2008. [Online]. Available: http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0172.pdf

[17] S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119." March 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt

Paper II

SLAs for Energy-Efficient Data Centres: The Standards-based Approach of the OPTIMIS project[2]

Wolfgang Ziegler

Fraunhofer Institute SCAI, 53754 Sankt Augustin, Germany
wolfgang.zieglerl@scai.fraunhofer.de

Abstract: In current Cloud environments customers aiming to select a provider that offers energy efficient infrastructure usually depend on believing in the providers' publicity. In general they have little chances to alter the standard contract the big providers are offering. Smaller providers might offer the possibility to include energy efficiency as a clause in their paper framework contract. However, so far there is no way to dynamically create a Service Level Agreement with a provider that includes a certain level of energy efficiency the provider guarantees. The European project OPTIMIS is focussing on optimisation of cloud infrastructure services meeting demands from service providers. Besides a number of other parameters like trust, risk, cost, data protection this also includes aspects of energy efficiency. We describe the standards-based approach of OPTIMIS for negotiating and creating Service Level Agreements between service providers and infrastructure providers regarding energy efficiency of a data centre.

[2]By permission of Springer Verlag

SLAs for Energy-Efficient Data Centres: The Standards-Based Approach of the OPTIMIS Project

Wolfgang Ziegler

Fraunhofer Institute SCAI, 53754 Sankt Augustin, Germany
wolfgang.zieglerl@scai.fraunhofer.de

Abstract. In current Cloud environments customers aiming to select a provider that offers energy efficient infrastructure usually depend on believing in the providers' publicity. In general they have little chances to alter the standard contract the big providers are offering. Smaller providers might offer the possibility to include energy efficiency as a clause in their paper framework contract. However, so far there is no way to dynamically create a Service Level Agreement with a provider that includes a certain level of energy efficiency the provider guarantees. The European project OPTIMIS is focussing on optimisation of cloud infrastructure services meeting demands from service providers. Besides a number of other parameters like trust, risk, cost, data protection this also includes aspects of energy efficiency. We describe the standards-based approach of OPTIMIS for negotiating and creating Service Level Agreements between service providers and infrastructure providers regarding energy efficiency of a data centre.

1 Introduction

The European project OPTIMIS is addressing optimisation of cloud infrastructure services meeting demands from service providers (SP), e.g. when public and private Clouds are federated in different configurations. This optimisation considers trust, risk, eco-efficieny, cost (the TREC parameters), data protection and data security (as presented in [10]. We describe the standards-based approach of OPTIMIS for negotiating and creating Service Level Agreements between end-users or service providers and infrastructure providers regarding energy efficiency of a data centre.

So far, dynamic electronic Service Level Agreements (SLAs) are rarely used in Clouds to define and agree upon the QoS the infrastructure provider (IP) will deliver to its customers regardless whether the customer is an end-user or a SP providing its services to the end-user. Even worse, the standard SLA offered by providers covers resource properties and an expected uptime but does not include any aspects of energy efficiency of a data centre a customer could claim. Nevertheless, a number of activities in different standards bodies have produced standards or are on the way to produce a standard relevant in the area of Cloud

J. Huusko et al. (Eds.): E^{2}DC 2012, LNCS 7396, pp. 37–46, 2012.

computing. The most relevant for SLAs on energy efficiency are briefly described in this paper along with those that could play a role when further extended. The focus of the following sections is the descriptions of the standards-based approach of OPTIMIS for negotiating and creating SLAs between IPs and SPs that include the energy efficiency level agreed upon between customer and provider. The OPTIMIS developments enable the customer to reach a binding agreement with the provider and support the provider in optimising its infrastructure with respect to energy efficiency.

The remainder of the paper is organised as follows. Section 2 presents related work. Section 3 highlights current activities in different standards bodies regarding Cloud-relevant standards. Section 4 discusses general aspects of Service Level Management (SLM), how SLM is used in OPTIMIS and the technology for creating the Service Level Agreements. Section 5 presents the life-cycle of the OPTIMIS SLAs. A discussion of the approaches followed in the project regarding integration of energy efficiency in the SLAs can be found in the following Section 6 presenting a static approach based on external certification and an approach based on calculated average rates in Section 6.1 and 6.2 respectively along with the corresponding term language in 6.3 and gives an outlook on work started recently to integrate real-time power consumption monitoring data into the OPTIMIS environment in 6.4. The paper concludes with a summary and plans for future work.

2 Related Work

Until now only limited research has been focusing on SLAs for Clouds. In [14] an approach for using SLA in a single cloud scenario is presented. In case of SLA violation a penalty mechanism is activated rather than dynamically extending the hardware resources provided as realised in OPTIMIS. Moreover, despite the fact the authors complain about missing standards in this area they propose using IBM's proprietary solution developed 7 years ago. In 2010 [4] addresses challenges for IaaS, identifying Service Level Agreements as one of them which attracted little attention so far. IBM recently published a review and summary of cloud service level agreements [8] as part of their Cloud Computing Use Cases Whitepaper [9], which suggests SLA requirements and metrics for Cloud environments. However, this is not linked to any concrete implementation. Finally, in the European RESERVOIR project [15] SLAs have been used to define and monitor QoS of services deployed in Cloud infrastructures but no QoS parameters related to the infrastructure the services have been deployed in has been covered by these SLAs. As of today, to our best knowledge there is no related work on using Service Level Agreements between a user or a SP and an IP to establish an electronic contract regarding the required level of energy efficiency of a data centre.

3 Standardisation Approaches

Cloud technology has entered the focus of a number of standardisation bodies over the last three years, e.g. NIST, IEEE, TeleManagement Forum, OASIS, DMTF. However, most often the work is still in an early phase and far from delivering implementable specifications. Exceptions are, e.g. the OCCI [12] and WS-Agreement [1] specifications of the open Grid Forum (OGF). The latter is defining language and a protocol for creating SLAs, which was already published May 2007. While having been developed in the context of Grid computing WS-Agreement Specification is domain agnostic and allows the usage of any domain specific or standard condition expression language to define SLOs. In 2011 it was complemented by WS-Agreement Negotiation [2][3], which offers a multi-step negotiation protocol on top of WS-Agreement. The second broadly used standard of the OGF is the Open Cloud Computing Interface (OCCI), which is used by a number of Cloud providers and Cloud middleware stacks as an open, interoperable interface. The Distributed Management Task Force (DMTF) published in 2009 version 1.0 of the Open Virtualisation Format (OVF) [13] specification. OVF provides a standard format for packaging and describing virtual machines and applications for deployment across heterogeneous virtualisation platforms, while the related profiles standardise many aspects of the operational management of a heterogeneous virtualised environment. In OPTIMIS OVF is used to describe disk images. More recently the Telemanagement Forum has started as working group on Cloud SLAs. IEEE has launched a standards activity related to cloud federation with its group P3202. The group has no plans yet for using SLAs for the federation process. In 2011 National Institute of Standards and Technology (NIST) published the NIST Cloud Computing Reference Architecture [11] and the NIST Cloud Computing Standards Roadmap [7], however so far Cloud SLAs are not included. Moreover, the OASIS TC on Topology and Orchestration Specification for Cloud Applications (TOACA) published in March 2012 the Committee Specification Draft [16] for the formal description of service templates used to specify the topology and orchestration of IT services. A WS-Agreement rendering is under discussion. Finally, in April 2012 several of the before mentioned organisations started a cross SDO working group to better understand, coordinate and integrate their respective work related to Cloud standards.

4 Service Level Management

OPTIMIS implements two phases for Service Level Management (SLM): (i) specification of the requirements an SP or its customer has with respect to the virtualised infrastructure, and (ii) negotiating and creating the SLA on these requirements between SP and IP.

SLAs represent contractual relationships between the SP and the IP. SLAs describe the infrastructure service that is delivered, the functional and non-functional properties of the service, and the obligations of each party involved.

Additionally, SLAs define guarantees for the functional and non-functional service properties. These guarantees specify a service level objective that must be met in order to fulfil a guarantee. Compensations in form of penalties and rewards may be linked with guarantees becoming due in case the guarantee is fulfilled or violated respectively. The OPTIMIS Cloud QoS (QoS) component responsible for the SLM is an extension of the WSAG4J framework [17], which is an implementation of WS-Agreement and WS-Agreement Negotiation realised at the Fraunhofer Institute SCAI. It provides comprehensive support for common SLA management tasks such as SLA template management, SLA negotiation and creation, and SLA monitoring and accounting.

5 Service Level Agreements

5.1 SLA Life-Cycle

The OPTIMIS SLAs have a fairly standard life cycle, which they are running through from the initial preparation of the templates for an agreement up the evaluation whether the agreement has been fulfilled, or partly or completely violated. SLA life-cycles typically look like decribed in [5] and consist of several phases; these phases are shown in Figure 1.

Fig. 1. SLA life-cycle

The activities in the individual phases can be described and mapped to the OPTIMIS toolkit components as shown below. Involved components are QoS, the SP's Service Deployment Optimiser (SDO) and the IP's Cloud Optimiser (CO).

1. *SLA Development:* In this phase the SLA templates are developed. In OPTIMIS generic templates have been defined and are accessible from the service providers. After selecting the service provider based on the requirements in the service manifest (SM) the QoS component extends the generic template with the SM for further negotiation.
2. *Negotiation:* In this phase the SLA is negotiated and the contracts are executed. The QoS component negotiates with the IP's CO component
3. *Implementation:* where the SLA is generated. Triggered by the SP through the SDO, done by the QoS component in OPTIMIS.
4. *Execution:* The SLA is executed, monitored, and maintained. The QoS component provides an interface to subscribe to an information service where, e.g. the service provider can retrieve monitoring information during this phase.

5. *Assessment:* Evaluation of the SLA performance. In this phase, a re-evaluation of the initial SLA template might be done. This is done by the QoS component, the result can be retrieved by other OPTIMIS components, e.g. the SDO.

Additional transitions between the *Implementation* or the *Execution* phase back to the *Negotiation* phase can be performed when re-negotiation of an agreement is required.

After the agreement has been created it is possible to retrieve monitoring information on the states of the agreement terms. Based on the evaluation of the individual service term states it can be decided whether an agreement is still fulfilled or risks to be violated if no appropriate countermeasures are taken. The sequence diagram in Figure 2 depicts the general steps required for querying monitoring information from an agreement. Please note: in the OPTIMIS environment the *ServiceConsumer* is the SP and the *ServiceProvider* is the IP.

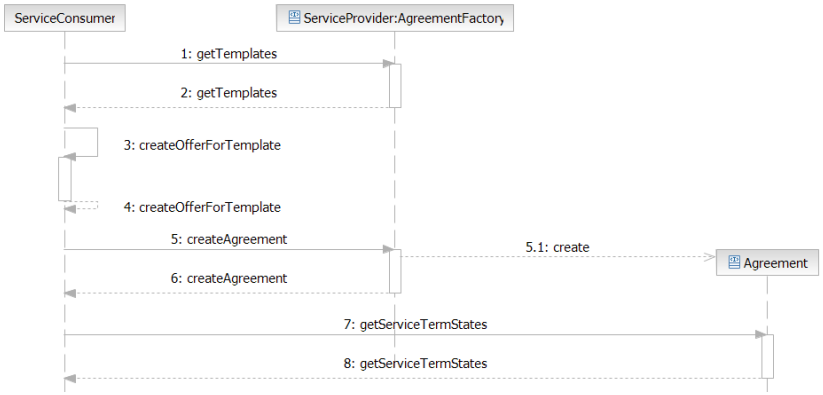


Fig. 2. SLA monitoring

6 Term Language for the Specification of Energy Efficiency Requirements

OPTIMIS follows three complementary approaches for assessing energy efficiency of the IP. The first year was focussing on certification regarding eco-efficiency. The current second year develops a model for breaking down the overall power consumption to average rates for Virtual Machines (VM) depending on classes of infrastructure the VMs are deployed to. The third year will finally develop prototypical infrastructure to integrate real-time monitoring of power consumption.

6.1 Approach Based on Static Certification Information

In order to provide SLA-aware infrastructure services with extended QoS capabilities the SLA management layer requires capabilities to detect under-performing

with respect to the SLAs defined. However, in the case of static information regarding energy efficiency of a data centre we rely on external certifications currently available. Provider selection is done based on these properties, no monitoring is needed when the services are deployed in the data centre as these properties are not expected to change during a deployment.

The core measurable for eco-efficiency are **Energy** (in kilowatts per hour per VM) and **Carbon** (in kg CO2 per hour per VM) associated with operational energy. In addition, there are a number of ways, beyond basic energy use, that datacenter operators can demonstrate eco-efficiency. The following already are used commonly:

- PUE (power usage effectiveness). This measures the infrastructure overhead of the facility and is a measure of the energy efficiency of the data centre (but not the IT equipment).
- Conformance to the EU Code of Conduct (COC) for data centers. This sets out best practices and operators must endorse the code or be a full signatory.
- Energy Star for data centres. This US driven certification of energy efficiency will be promoted across the world.
- LEED for data centres. This standard rates a datacenter for the positive impact on the environment.

Table 1 summarises the initial set of basic terms relating to eco-efficiency that are used in OPTIMIS SLAs.

Table 1. Basic parameters relating to eco-efficiency used in OPTIMIS SLAs

Terms and parameters	**Metric**
Is the centre European Code of Conduct for data centre compliant?	Yes/No
Does the centre have an Energy Star for data centre rating?	Points range or star rating, No
Is the centre LEED for data centre rated?	Platinum, Gold, Silver, Bronze, No

6.2 Approach Based on Availability of Specific Data Related to Energy Consumption and Carbon Footprint

For the second approach we are using the overall power consumption information of a data centre for a certain period (and corresponding CO2 equivalent based on the local carbon content factor and any offsets or credits etc) already available for the infrastructure provider through the bills received from its electricity supplier. If not published, the CO2 equivalent can be estimated from the region a data centre is located and the power supplier's technology used for electricity generation. Other sources for this information include, e.g. the annual reports of Greenpeace on data centres [6]. Based on this information further estimates

can be made by the infrastructure provider to break the energy consumption and the CO2 emission down to an average rate per hour per VM instance. Of course, these calculations and their results also will need a certification by an independent institution to increase trustworthiness for customers. Providing this information to customers or brokers will allow taking the average rates into account when choosing the infrastructure provider. Further, the rates specified as requirements in the manifest become part of the SLA enabling the provider to check against internal measurements at run-time and to take counter measures in case of exceeding the values agreed upon. Corrective measures could be taken locally by the IP to avoid that the SLA is violated or - in case the provider realises that local countermeasures won't solve the problem - by migrating the service to another provider (after negotiating a new agreement that provides at least the same QoS level as the original SLA with the customer). Of course, initially only IPs that are part of the OPTIMIS consortium are ready to publish their average rates. However, we expect the situation will change over time when energy efficiency becomes a competitive factor for the IPs and SLAs with customers will include the requested energy efficiency level.

6.3 OPTIMIS Service Manifest

As mentioned before, we use the SM to describe the requirements regarding service QoS from the viewpoint of the SP or end-user. The following code exemplifies the part of the SM related to energy efficiency. The section consists of two parts. The first part specifies the requirements regarding static certification of an infrastructure provider as described in 6.1. The second part specifies requirements can be used used for infrastructure providers periodically publishing information allowing to assess their energy efficiency.

```
<opt:EcoEfficiencySection xmlns:opt="http://schemas.optimis.eu/optimis/">
    <opt:LEEDCertification>NotRequired</opt:LEEDCertification>
    <opt:BREEAMCertification>NotRequired</opt:BREEAMCertification>
    <opt:EuCoCCompliant>false</opt:EuCoCCompliant>
    <opt:EnergyStarRating>100</opt:EnergyStarRating>
```

In the section above the infrastructure provider's customer requests that the data-centre should have an Energy Star rating of 100. This could be verified before creating the SLA by accessing the Energy Star website, which would list the data centre once it has achieved the certification. However, currently there are no data centres listed there.

```
    <opt:EnergyConsumptionSection>
        <opt:AverageRatesPerVM>
            <opt:Large>
                <opt:EnergyConsumptionPerVm>
                    <opt:Electricity opt:unit="kWh">300</opt:Electricity>
                    <opt:Carbon opt:unit="kg">150</opt:Carbon>
                </opt:EnergyConsumptionPerVm>
```

```
            </opt:Large>
            <opt:Medium>
                <opt:EnergyConsumptionPerVm>
                    <opt:Electricity opt:unit="kWh">200</opt:Electricity>
                    <opt:Carbon opt:unit="kg">100</opt:Carbon>
                </opt:EnergyConsumptionPerVm>
            </opt:Medium>
            <opt:Small>
                <opt:EnergyConsumptionPerVm>
                    <opt:Electricity opt:unit="kWh">100</opt:Electricity>
                    <opt:Carbon opt:unit="kg">50</opt:Carbon>
                </opt:EnergyConsumptionPerVm>
            </opt:Small>
        <opt:AverageRatesPerVM>
    </opt:EnergyConsumptionSection>
```

The section above specifies exemplifies the required average rates for the three types of VMs (large, medium, small, similar to Amazon's classification) plus the respective CO2 equivalent for a concrete data centre. A SP or a broker may use the average rates provided by different IPs to select the provider and the type of VM best suited for the service to be deployed while trying to minimise the power consumption and corresponding CO2 equivalent according to the request in the SM.

6.4 Approach Based on Availability of Real-Time Monitoring Data

For the third approach we will be using real-time monitoring information regarding actual power consumption (and the calculated corresponding local CO2 equivalent) of the infrastructure a service is deployed to. It should be noted that while the average power consumption per hour and VM could be similar across different data centres the local CO2 equivalent may differ significantly since it heavily depends on the providers' energy choices.

Members of the OPTIMIS consortium are currently equipping machines in two testbeds with off-the-shelf electricity meters to figure out how real-time measurements can be retrieved and stored in the monitoring database in a useful way. On the other hand, many providers are already gathering power consumption data, which is used internally to monitor and optimise infrastructure and cooling facilities. Having access to this data will allow to check against the values in the SLA and to take counter measures in case of exceeding the values agreed upon. Corrective measures could be taken by the IP to avoid that the SLA is violated or - in case the provider cannot fulfil the SLA - either by the IP or by the SP, e.g. migrating the service to another provider (after negotiating a new agreement). Initially, only a few IPs will be willing to expose their internal power monitoring to their customers. However, as with the second approach we expect the situation will change over time when energy efficiency becomes a competitive factor for the IPs and SLAs with customers will include the requested energy efficiency level. Requirements based on the availability of real-time monitoring

data of power consumption will be included in manifest and SLAs in the third project year starting summer 2012.

7 Conclusions and Future Work

We presented the European OPTIMIS project's standards-based approach for SLM in a multi-cloud environment. The focus of the work described is on SLAs between end-user or SP and IP for user driven selection of data centres based on their energy efficiency and contractual guarantees. The SLA technology is based on WS-Agreement and WS-Agreement Negotiation using the Java implementation WSAG4J. Next steps of the OPTIMIS project are full integration with the cloud-providers' real-time monitoring of the energy consumption. In parallel we are preparing the publication of the term languages developed in OPTIMIS in the document series of the Open Grid Forum as an initial step to converge to standardised term languages. The SLA framework is already available for download [17]. The first version of the integrated OPTIMIS toolkit is available through the OPTIMIS website, yearly updates reflecting the progress of the toolkit will be published there.

Acknowledgements. Work reported in this paper has been co-funded by the European Commissions ICT programme in the FP7 project OPTIMIS under grant #257115 and by the German Federal Ministry of Education and Research in the D-Grid project under grant #01AK800A.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). REC Recommendation, Open Grid Forum (2011), GFD.192 recommendation, `http://www.ogf.org/documents/GFD.192.pdf`
2. Battré, D., Brazier, F.M.T., Clark, K.P., Oey, M., Papaspyrou, A., Wäldrich, O., Wieder, P., Ziegler, W.: A proposal for WS-Agreement Negotiation. In: IEEE Grid 2010 Conference (2010)
3. Battré, D., Brazier, F.M.T., Clark, K.P., Oey, M., Papaspyrou, A., Wäldrich, O., Wieder, P., Ziegler, W.: Web Services Agreement Negotiation Specification (WS-Agreement Negotiation). P-REC Proposed Recommendation. Open Grid Forum (2011), GFD.193 proposed recommendation, `http://www.ogf.org/documents/GFD.193.pdf`
4. Dawoud, W., Takouna, I., Meinel, C.: Infrastructure as a service security: Challenges and solutions. In: 2010 The 7th International Conference on Informatics and Systems (INFOS), pp. 1–8 (March 2010)
5. Sun, W., et al.: The Role of XML in Service Level Agreements Management. Network Management Research Center, Beijing. Jiaotong University, IEEE (2005)
6. How Clean is Your Cloud? Greenpeace report (April 2012), `http://www.greenpeace.org/international/Global/international/publications/climate/2012/iCoal/HowCleanisYourCloud.pdf`

7. Hogan, M., Liu, F., Sokol, A., Tong, J.: NIST Cloud Computing Standards Roadmap – Version 1.0. Technical report, National Institute of Standards and Technology: NIST Cloud Computing Standards Roadmap Working Group (2011)
8. Review and summary of cloud service level agreements. Accessible on IBM's web-site, http://public.dhe.ibm.com/software/dw/cloud/library/cl-rev2sla-pdf.pdf
9. Cloud Computing Use Cases Whitepaper Version 4.0., http://www.scribd.com/doc/18172802/Cloud-Computing-Use-Cases-Whitepaper
10. Lawrence, A., Djemame, K., Wäldrich, O., Ziegler, W., Zsigri, C.: Using Service Level Agreements for Optimising Cloud Infrastructure Services. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 38–49. Springer, Heidelberg (2011)
11. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: NIST Cloud Computing Reference Architecture. Technical report, National Institute of Standards and Technology: Information Technology Laboratory (2011)
12. Metsch, T., Edmonds, A., Papaspyro, A.: Open Cloud Computing Interface - Core. P-REC Proposed Recommendation, Open Grid Forum (2011) GFD.183 proposed recommendation, http://www.ogf.org/documents/GFD.183.pdf
13. Open Virtualization Format Specification Version 1.1.0. Accessible on DMTF's web-site, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf
14. Patel, P., Ranabahu, A., Sheth, A.: Service Level Agreement in Cloud Computing. In: Cloud Workshops at OOPSLA 2009 (2009)
15. RESERVOIR Project Web-Site. Project Website, http://62.149.240.97/
16. Topology and Orchestration Specification for Cloud Applications Version 1.0. OASIS Committee Specification Draft 01 (March 08, 2012), http://docs.oasis-open.org/tosca/TOSCA/v1.0/csd01/TOSCA-v1.0-csd01.html
17. WSAG4J - Web Services Agreement for Java. Web site (March 18, 2012), http://packcs-e0.scai.fraunhofer.de/wsag4j

Paper III

Legal Restraints and Security Requirements on Personal Data and Their Technical Implementations in Clouds[3]

Benno Barnitzke[1], Wolfgang Ziegler[2], George Vafiadis[3], Srijith Nair[4], George Kousiouris[3],Marcelo Corrales[1], Oliver Wäldrich[2], Nikolaus Forgó[1], Theodora Varvarigou[3]

[1]Institut für Rechtsinformatik, Leibniz Universität Hannover
Königsworther Platz , 30167 Hannover, Germany
Tel: +49 511 762-8242, Fax: +49 511 762-8290
Email: { barnitzke, corrales, forgo}@iri.uni-hannover.de

[2]Fraunhofer-Institute for Algorithms and Scientific Computing (SCAI)
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany
Tel: +49 2241 14 2258; Fax: +49 2241 14 42258
Email: {Wolfgang.Ziegler, oliver.waeldrich}@scai.fraunhofer.de

[3]National Technical University of Athens, 9 Iroon Polytechniou Str., Athens, 15773, Greece
Tel:+302107722546, Email: gvaf@iccs.gr, gkousiou@mail.ntua.gr, dora@telecom.ntua.gr

[4]BT Innovate & Design, Martlesham Heath, Ipswich IP5 3RE, United Kingdom
Tel : +44 1473 606243, Email: srijith.nair@bt.com

Abstract: Cloud computing has emerged as the new trend in the IT industry. However, in order for the adopters of this technology to be legally compliant with regard to the handling of personal data, a series of actions must be undertaken. In this paper, we present the legal requirements that exist inside the EU with regard to transnational data transfer and storage. Based on these legal requirements, we then used the WS-Agreement standard to create corresponding SLAs. Subsequently, we extended existing and well known technologies in order to create a data management framework that is necessary from an Infrastructure Provider point of view, so that the latter can be considered as a trusted entity with regard to data management. As a result, customers and Cloud providers using

[3]By permission of IIMC

the presented SLA and data management framework are able to be compliant with the legal requirements stipulated by the Data Protection Directive with regard to transnational data transfers.

eChallenges e-2011 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2011
ISBN: 978-1-905824-27-4

Legal Restraints and Security Requirements on Personal Data and Their Technical Implementation in Clouds

Benno BARNITZKE[1], Wolfgang ZIEGLER[2], George VAFIADIS[3], Srijith NAIR[4], George KOUSIOURIS[3], Marcelo CORRALES[1], Oliver WÄLDRICH[2], Nikolaus FORGÓ[1] and Theodora VARVARIGOU[3]

[1] *Institut für Rechtsinformatik, Leibniz Universität Hannover, Königsworther Platz , 30167 Hannover, Germany*
Tel: +49 511 762-8242, Fax: +49 511 762-8290,
Email: { barnitzke, corrales, forgo}@iri.uni-hannover.de

[2]*Fraunhofer-Institute for Algorithms and Scientific Computing (SCAI), Schloss Birlinghoven, D-53754 Sankt Augustin, Germany*
Tel: +49 2241 14 2258; Fax: +49 2241 14 42258,
Email: {Wolfgang.Ziegler, oliver.waeldrich}@scai.fraunhofer.de

[3]*National Technical University of Athens, 9 Iroon Polytechniou Str., Athens, 15773, Greece*
Tel:+302107722546, Email: gvaf@iccs.gr, gkousiou@mail.ntua.gr, dora@telecom.ntua.gr

[4]*BT Innovate & Design, Martlesham Heath, Ipswich IP5 3RE, United Kingdom,*
Tel : +44 1473 606243, Email: srijith.nair@bt.com

Abstract: Cloud computing has emerged as the new trend in the IT industry. However, in order for the adopters of this technology to be legally compliant with regard to the handling of personal data, a series of actions must be undertaken. In this paper, we present the legal requirements that exist inside the EU with regard to transnational data transfer and storage. Based on these legal requirements, we then used the WS-Agreement standard to create corresponding SLAs. Subsequently, we extended existing and well known technologies in order to create a data management framework that is necessary from an Infrastructure Provider point of view, so that the latter can be considered as a trusted entity with regard to data management. As a result, customers and Cloud providers using the presented SLA and data management framework are able to be compliant with the legal requirements stipulated by the Data Protection Directive with regard to transnational data transfers.

1. Introduction

Cloud Computing involves location-independent processing and leads to the global transfer of personal data. On the one hand, the scaling of resources in the cloud enables users to save considerable costs. On the other hand, data protection laws might prevent cloud providers from transferring the data to other countries and additionally requires them to implement data security measures. Cloud providers therefore need smart tools to ensure compliance both with data export and data security requirements and thus extend their business to applications dealing with personal data.

Due to dynamic resource allocations enabling cloud users to scale their applications across different cloud providers, cloud computing entails a highly distributed architecture. This implies more data in transit than in traditional infrastructures [1] and even leads to massive and worldwide transfers of personal data within a cloud. However, international data transfers to a country outside the European Economic Area (EEA) are subject to clear

legal restrictions. Users and cloud providers therefore facethe problem of ensuring that personal data is not being transmitted to these countries without further legal safeguards.

We will describe these constraints in more detail and provide legal advice on a technical solution, enabling both users and providers to comply with these rules. This solution could be used by Service Providers (SPs) or Infrastructure Providers (IPs) who would like to offer data protection compliant cloud services within the EU. This way, SPs can offer cloud enabled applications (i.e. CRM software) running on a location aware cloud data management system without taking the risk of illegitimately transferring personal data to third countries.

2. Objectives

This paper aims to highlight the most relevant legal issues of cloud computing on a European level, regarding the transfer of personal data to third countries. It also provides a solution as to how to express these requirements on a technical level in the SLAs and the data management system. We show how these legal requirements are expressed in the SLA and what action is needed in order to include the information about the location of the data processing into the data management system. Furthermore, we address data security issues, particularly which technical measures in the cloud architecture should be implemented to protect personal data against unauthorised processing. A technology framework has been created in order to implement the management of these requirements and additions to existing toolkits. Certain approaches are discussed in order to ensure that the overall goal will be addressed.

The major objectives of the paper are the following:

1. Detailed analysis of the legal requirements based on the current European legal data protection framework. This will aid Cloud providers in understanding what is legally expected from them within the European Economic Area.
2. Contribution to existing specifications (e.g. regarding SLAs) in order to includeterms and conditions that are related to step 1.
3. Proposal of a technical framework that can be used by IPs in order to implement the requirements produced by step 1 and, as a result, be compliant with current data protection legislation.

3. Methodology

Firstly, this paper outlines the legal conditions for data transfers and storage in clouds outside the EEA. We consider legal requirements as part of the Quality of Service (QoS) a Cloud user requires from its Cloud provider. The placement of data is fixed by a binding Service Level Agreement (SLA) with the cloud provider. A specification for describing these legally binding terms is included, in order to externalise resources used by an IP.

Finally, we describe the measures that are needed in order for the infrastructures to comply with the requirements by using existing tools and technologies from the OPTIMIS project (www.optimis-project.eu). In order to implement these, we extended existing toolkits that are widely used and adopted.

4. Analysis of Legal Requirements

4.1 Data Protection Requirements Regarding Transnational Data Transfers

Within the territorial scope of Directive 95/46/EC (the Data Protection Directive, hereinafter referred to as DPD), no restrictions exist for data transfers to countries within the EEA, since all Member States are deemed to provide an adequate level of protection [2]. The territorial scope of the DPD comprises all 27 EU Member States plus the European Economic Area (EEA), thus including Iceland, Liechtenstein and Norway. As a consequence, data transfers within the territorial scope of the Data Protection Directive are allowed. Furthermore, the European Commission has explicitly stated for specific countries outside the EU (after thorough examination of their national laws) that they also provide an adequate level of protection. These countries are Switzerland, Canada, Argentina, Guernsey, Isle of Man, Israel (since 31 January 2011) and US organisations that take part in the US safe harbour program [3]. While transfers to cloud providers in any of these countries are legitimate, transfers to any other country ('third countries') not mentioned here are prohibited if there are no further legal safeguards implemented by the data controller. The list of countries to which transfers are possible are therefore similar to a 'white list'. Thus, a cloud provider processing personal data must provide the necessary technical measures to prevent any transfer of data to third countries (= countries not listed on the 'white list') if no additional safeguards shall be taken (see Figure 3 for a graphical overview of admissible transfers of personal data). Although the transfer of personal data to these third countries is not allowed, there are additional legal bases (mostly on a contractual level) that provide an adequate level of protection for data exports even if the third country itself does not. According to Art. 26 (4) DPD, the European Commission may decide that certain standard contractual clauses offer sufficient safeguards for such kinds of data transfers. To this end, the Commission has set up a standardised set of clauses which can be used as a legal basis for transfers from each Member State to any third country (Standard Contractual Clauses) [4]. If a controller located within the EU or EEC enters into a contract which includes the EU Standard Contractual Clauses, the controller located outside the EU or EEC is considered to provide an adequate level of protection [5]. Consequently, if SLAs agreed between an end user and a cloud provider apply the (unmodified) EU Standard Contractual Clauses in a legally binding way, the transfer would be lawful. WS-Agreement should therefore offer cloud customers the possibility to include EU Standard Contractual Clauses. Otherwise, the data management system should restrict data placement to countries not included in the 'white list'. In view of these explanations, we derived three legal requirements for the distributed file system used in the OPTIMIS project (Table 1).

Table 1: Legal requirements for the Data Management System in the OPTIMIS cloud.

Provide information to end users about their data storage locations	**Support country-specific location for placement of personal data and separation from other data**	**Prohibit placement of data in third countries**
Cloud providers should disclose the fact of whether they are operating data centres in third countries or whether they are in a federation with other cloud providers that operate data centres in third countries.	The file system run in the cloud must be able to support locations, thus be "aware" where the data centres are located in order to enable users or cloud providers to decide where personal data may be transferred to. However, some users may process information	Where a user and cloud provider have not agreed upon additional safeguards for data transfers to third countries, the file system must not transfer personal data to data centres located in these countries. Requests to transfer data to locations in third

The user should be able to constantly track the location of the data processing by means of a monitoring tool.	not relating to an individual. Data of this kind does not fall under the scope of the DPD. Separation of the two categories of data would significantly reduce the impact on performance and management load for both users and IPs.	countries must therefore be denied by the Data Management System. However, where users and/or cloud providers have agreed to use Standard Contractual Clauses, the Data Manager should accept the transfer.

4.2 Data Security Requirements Regarding Transfer and Storage of Personal Data

The DPD also contains constraints with regard to the security of processing. As the recent Amazon EC2 outage has clearly shown [6], data loss or even destruction of personal data is a major concern in clouds. According to Art. 17 DPD, the controller must implement appropriate technical and organisational measures to protect personal data against accidental or unlawful destruction or accidental loss.

While "destruction" of personal data represents the complete removal or serious corruption of physical data (i.e. on the hard disk or in main memory) in such a way that their recovery is impossible, "loss" refers to unplanned events such as natural disasters or hardware failures. Consequently, the data in virtual machines should be backed up by replication on different physical machines in different data centre locations on a regular basis. Compliance with this provision is not only a legal requirement, but also in the best business interests of the cloud provider, as it helps to preserve company reputation and customer trust [1]. However, it is not solely the cloud provider's duty to protect personal data. Maintaining data integrity and availability is also an obligation of the cloud customer, which could mean running services across multiple providers [7]. To increase redundancy in cloud services, IaaS Providers should create a standard for sharing VM instances across clouds to simplify compliance with Art. 17 DPD. With the OPTIMIS toolkit being developed, this will – to some extent – come true, as it facilitates provisioning of services in Federated and Multi-cloud scenarios.

Due to considerable transfer and storage of personal data in clouds, as well as the aspect of multi-tenancy, confidentiality is also key for cloud computing. The DPD requires controllers and processors to protect personal data against unauthorised disclosure, in particular where the processing involves the transmission of data over a network. Hence, we recommend implementing strong encryption whenever an IP moves data within the cloud. Finally, we recommend that data should be encrypted at rest (stored data) with the same diligence as data determined to be transmitted, as they are exposed to the risk of disclosure in the same way. The DPD does not mention technical details of encryption type or strength. This depends on the type and quantity of data to be processed. A security analysis should be made in any case and a policy set in place. Compliance with that policy should be legally agreed on between all partners and should be constantly monitored.

5. Technology Description

As baseline technology for negotiating and creating the SLA selected, WS-Agreement, a standard of the Open Grid Forum (OGF), and WS-Agreement Negotiation are used. Besides the fact that WS-Agreement is a well-established standard, the rationale for using it is its flexibility. WS-Agreement, itself being completely domain agnostic, requires the use of a domain-specific term language to create SLAs for the domain. In OPTIMIS we develop term languages, e.g. for expressing Trust, Risk, Eco-efficiency and Costs (TREC

parameters) related to infrastructure services of a provider. In this context, we also define a language to express the requirements with respect to data transfer and security. These terms are then part of a legally binding SLA with the IP ruling the transfer of the data within its data centres. Since it is not possible to figure out whether the Service Description Terms (SDT) and related guarantees were fulfilled or violated without the ability to monitor the state of the system, we also are developing a monitoring instrument for data location tracking. Suitable XML schemas are described in order to fully capture the legal requirements that the infrastructure must take into consideration when accepting new data or enabling federation between different cloud providers.

Given that the above requirements are adequately expressed, what is left for the Cloud providers is to create the framework that will enable them to abide by their commitments agreed to in the SLAs. To this end, we will describe the design of the OPTIMIS project data infrastructure and the tools that are used in order to ensure that the end user's constraints are met, especially when two or more Cloud providers cooperate in the context of a federation.

This includes suitable RESTful [13] interfaces for the Data Manager to be informed of the constraints; to provide information regarding the current storage location of data; to have differentiated levels of security implementations as needed by the legal analysis; to provide information regarding the location of its datacentres etc. These interfaces must be coupled with the internal framework that keeps track of the current location of data and prevents their movement to domains that are restricted.

In the OPTIMIS project, the Hadoop Distributed File System (HDFS [14]) is used, in order to implement the storage system. On top of this implementation, the RESTful interfaces are applied, that directly manipulate the HDFS configuration, in order to guide the infrastructure. The metadata structures of HDFS are also used in order to notify and update the Cloud provider each time a data action is performed.

6. Developments – Inclusion of Legal Analysis in the Design

6.1 Creating SLAs by using WS-Agreement

As mentioned in section 3, we consider legal requirements as part of the QoS that can be requested for a Cloud infrastructure where services are going to be deployed. These QoS parameters are agreed upon between the SP and the IP after having negotiated an SLA. The dynamically created SLA, resulting from the negotiations, fixes the OPTIMIS TREC parameters and the legal requirements with respect to data location and encryption. Restraints and requirements discussed in the previous sections have been captured in a term language to be used in the electronic SLA.

For negotiating and creating SLAs, we use the WSAG4J framework developed at the *Fraunhofer Institute SCAI* [9]. WSAG4J is a full implementation of WS-Agreement [10]. Moreover, WSAG4J also provides an implementation of the current draft of the standard for negotiating SLAs based on WS-Agreement: WS-Agreement Negotiation [11].

For the restraints and requirements regarding data placement and data protection in the Cloud we use a schema based on OVF (the Open Virtualisation Format standard of the DMTF) [12]. The following XML-code fragments show a part of the schema (Figure 1) and a section of the corresponding template (Figure 2) where restrictions of data placement and the requirements with respect to encryption of the data are described.


```
<xs:simpleType name="DataProtectionLevelType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="DPA"/>
                <xs:enumeration value="None"/>
        </xs:restriction>
</xs:simpleType>
[…]
<xs:complexType name="EncryptionLevelType">
        <xs:choice>
                <xs:sequence>
                        <xs:element name="EncryptionAlgoritm" type="opt:EncryptionAlgoritmType"/>
                        <xs:element name="EncryptionKeySize" type="xs:int" default="128" minOccurs="0"/>
                </xs:sequence>
                <xs:sequence>
                        <xs:element name="CustomEncryptionLevel" type="xs:anyType"/>
                </xs:sequence>
        </xs:choice>
</xs:complexType>
```

Figure 1: Schema definition of the data protection requirements

```
<ws:ServiceDescriptionTerm ws:Name="DataConstraints" ws:ServiceName="MultipleImages">
      <opt:DataProtectionSection>
        <opt:DataProtectionLevel>DPA</opt:DataProtectionLevel>
        <opt:DataEncryptionLevel>
          <opt:EncryptionAlgoritm>AES</opt:EncryptionAlgoritm>
        </opt:DataEncryptionLevel>
      </opt:DataProtectionSection>
    </ws:ServiceDescriptionTerm>
```

Figure 2: SLA template for the legal data protection requirements

While the dynamic SLA created between SP and IPs in conjunction with a written ("paper") framework contract can provide binding guarantees for the two parties, additional verification of whether the terms of the SLA are fulfilled or violated is required. In the first implementation of the OPTIMIS toolkit, monitoring is already available for the TREC parameters. Monitoring of fulfilment of legal requirements will be implemented in the next iteration. Clearly, the most challenging aspect already identified is the automatic verification of the location of the datacentre storing the data.

6.2 Design of a location-aware Data Management System

The Data Manager Storage System is the key element to achieve compliance with the already mentioned legal requirements. By using the Management Storage System, cloud-enabled applications such as CRM will be able to handle data in a data protection compliant way, even if the application itself does not support such a feature.

The challenges we face with regard to data management and data security in a federation have to be resolved in a practical and secure way. The simple case of a single IP having total control of all data nodes does not apply to a federated cloud. In this extended case we have multiple IPs sharing data among each other. The initial IP uses federated resources that may be placed in different geographical locations. Given that the legal requirements are correctly expressed in the SLA, the OPTIMIS Data Manager (DM) is able to receive them and act upon them accordingly, by mediating the HDFS placement policy

and demanding that the blocks of personal data are kept within the geographical domain specified by the end user. To this end, we take advantage of HDFS's rack awareness characteristics in order to ensure that during runtime the relevant policies are followed. Furthermore, replication aspects are handled through HDFS's interfaces and can be manipulated so that the necessary reliability can be achieved as demanded by the legal requirements. For example, replicas may be placed in different racks or even in different datacentres so that no single point of failure exists, neither at the rack nor at the geographical location level.

However, in order to be proactive, the DM of the IP also exposes a RESTful interface that provides an XML description of the location of its datacentres. This is necessary during the selection process performed by the SP. Thus the SP can filter out the IPs that cannot meet the legal requirements. Furthermore, during the initial account creation by the SP on the IP Data Manager, a flag is used in order to indicate whether the specific service contains personal data as defined in the Data Protection Directive and thus needs the advanced data location management offered by the DM.

Furthermore, the DM will expose a GUI to end users, through which the latter will be able to monitor at any time the location of their data and the history of their transfers to other locations (e.g. in the case of federation or load balancing decisions). More information on the details of OPTIMIS data management can be found in [15]. In Figure 3 we provide a graphical overview of the Data Management System.

6.3 Security measures regarding the confidentiality of personal data during storage and transfer phase

OPTIMIS IPs provide the customers with a secure storage device that can be used to store sensitive information. This storage device is encrypted at block device level and in real time, providing seamless transparent secure storage that can be used by all applications running in the customer's virtualised environment. The keys to decrypt the storage device can be stored outside the virtualisation infrastructure of the IP, with even the option of hosting it within the SP's infrastructure. The key management server that is responsible for releasing the keys can be instructed to do so based on policy rules associated with the environment of the VM, like the OS, the CPU architecture, IP address etc.

Data moved between the VM and the storage is protected by using the SSHFS protocol which allows for application level encryption on top of the secure storage provided. This protects the data from snooping and other eavesdropping attacks during transfer.

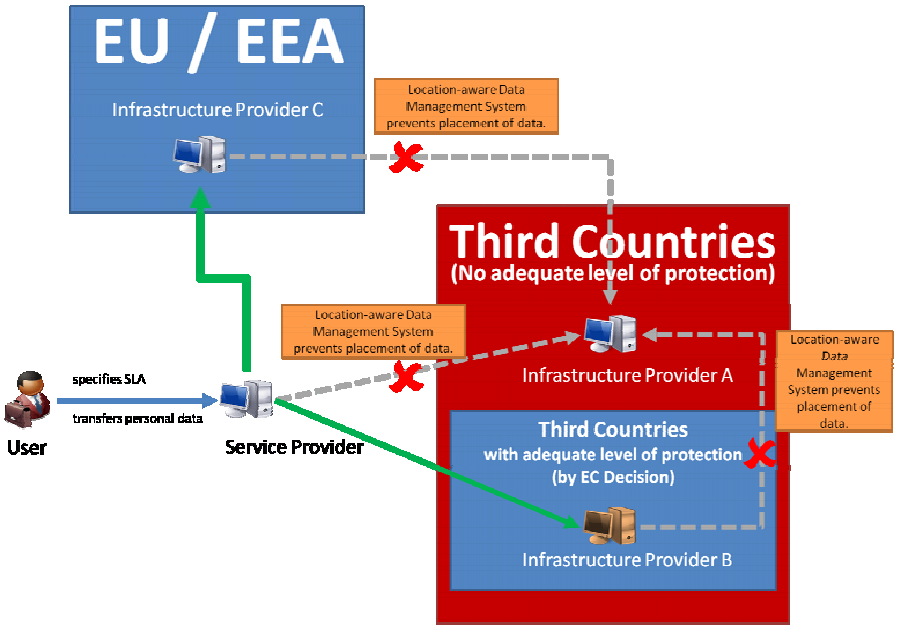


Figure 3: Graphical overview of Data Management System in OPTIMIS according to legal requirements of the Data Protection Directive.

7. Conclusions

Data processing involving personal data within a cloud is subject to clear restrictions with regard to countries not providing an adequate level of protection. For this purpose, we created an SLA framework by using WS-Agreement to let customers express where personal data may be transferred to, according to the legal requirements pursuant to the DPD. The Data Manager System is able to receive these requirements and distribute the data correspondingly.

The user should be able to constantly track the location of the data processing location by means of a monitoring tool. The distributed file system used in the cloud must support country-specific location for placement of personal data and prohibit placement of data in countries other than within the territorial scope of the DPD in case additional legal safeguards (Standard Contractual Clauses, Binding Corporate Rules) are not agreed in the SLA. Users processing data not relating to an individual should be able to separate their data to profit from improved performance and lower costs.

As regards legal data security obligations, data in clouds should always be replicated and stored redundantly elsewhere to avoid a Single Point of Failure. In order to protect the confidentiality and integrity of the data as required by the DPD, both transmission and storage of personal data demand efficient encryption.

In the OPTIMIS project, the aforementioned requirements are actively taken into consideration, in order to create a toolkit that will enable Cloud providers to abide by legal requirements in all modern scenarios of Cloud infrastructure usage.

Finally, it should be noted that globalisation in information management can have significant benefits with regard to cost or reliability of the infrastructures, but the fact remains that the current legal requirements within the EU prevent these benefits from being fully utilised. By using a technical framework that corresponds to these requirements, European Cloud customers and Cloud providers will be able to exploit cloud infrastructures that reside in different continents, while remaining compliant with European data protection

legislation. Furthermore, they can extend their business pool to cases or applications that process personal data.

Acknowledgement

This research is partially funded by the European Commission as part of the European IST 7th Framework Programme through the project OPTIMIS under contract number 25715.

References

[1] European Network and Information Security Agency (ENISA) 2009. Cloud Computing – Benefits, risks and recommendations for information security [online]. Available at: http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport [Accessed 20 April 2011], p. 38.
[2] Kuner, C., 2007. European Data Protection Law: Corporate Compliance and Regulation. 2nd ed. Oxford: Oxford University Press, p. 153.
[3] For further information see Commission decisions on the adequacy of the protection of personal data in third countries [online]. Available at: http://ec.europa.eu/justice/policies/privacy/thridcountries/index_en.htm [Accessed 20 April 2011].
[4]Commission Decision 2010/87/EU of 5 February 2010 on standard contractual clauses for the transfer of personal data to processors established in third countries under Directive 95/46/EC of the European Parliament and of the Council.
[5] Helbing, T. How the New EU Rules on Data Export Affect Companies in and outside the EU [online]. Available at: http://www.thomashelbing.com/en/how-new-eu-rules-data-export-affect-companies-and-outside-eu [Accessed 20 April 2011].
[6] Von Eicken, T. Amazon EC2 outage: summary and lessons learned, RightScale Blog [online]. Available at http://blog.rightscale.com/2011/04/25/amazon-ec2-outage-summary-and-lessons-learned/ [Accessed 26 April 2011].
[7] Evans, C. So Your AWS-based Application is Down? Don't Blame Amazon [online]. Available at http://www.thestoragearchitect.com/2011/04/22/so-your-aws-based-application-is-down-dont-blame-amazon/ [Accessed 26 April 2011].
[8] Finley, K. Stop Blaming the Customers - the Fault is on Amazon Web Services [online]. Available at http://www.readwriteweb.com/cloud/2011/04/almost-as-galling-as-the.php [Accessed 26 April 2011].
[9] WSAG4J – WS-Agreement for Java [online]. Available at http://packcs-e0.scai.fraunhofer.de/wsag4j
[10] WS-Agreement – Web Services Agreement [online]. Available at http://www.ogf.org/documents/GFD.107.pdf.
[11] WS-Agreement Negotiation – Web Services Agreement Negotiation [online]. Available at https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.graap-wg/docman.root.current_drafts.ws_agreement_negotiation_specifi/doc16194.
[12] OVF - Open Virtualization Format [online]. Available at http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.0.0.pdf.
[13] Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern Web architecture. ACM Trans. Internet Technol. 2, 2 (May 2002), 115-150. DOI=10.1145/514183.514185 http://doi.acm.org/10.1145/514183.514185.
[14] http://hadoop.apache.org/common/docs/current/hdfs_design.html.
[15] OPTIMIS Project D1.2.2.1 (MD1) OPTIMIS Detailed Design, UMEA and other partners, November 2010.

Paper IV

SLA-based management of software licenses as web service resources in distributed computing environments[4]

Claudio Cacciari[a], Daniel Mallmann[b], Csilla Zsigri[c], Francesco D'Andria[d], Björn Hagemeier[b], Angela Rumpl[e], Wolfgang Ziegler[e], Josep Martrat[d]

[a]CINECA, 40033 Casalecchio di Reno, Italy
[b]Forschungszentrum Jülich GmbH, 52428 Jülich, Germany
[c]The 451 Group, London, WC1E6HH, United Kingdom
[d]Atos Origin - Research and Innovation, Barcelona, 08011, Spain
[e]Fraunhofer Institute SCAI, Department of Bioinformatics, 53754 Sankt Augustin, Germany

Abstract: Until recently the use of applications requiring a software license for execution was quite limited in distributed environments. Due to the mandatory centralised control of license usage at application runtime, e.g. heartbeat control by the license server running at the home site of a user, traditional software licensing practices are not suitable especially when the distributed computing infrastructure stretches across administrative domains. In this paper we present a novel approach for managing software licenses as web service resources in distributed service oriented environments. Licenses become mobile objects, which may travel to the environment where required to authorise the execution of a license protected application. A first implementation has been realised for dynamic Grid environments in the European SmartLM project co-funded by the European Commission. The SmartLM solution decouples authorisation for license usage from authorisation for application execution. All authorisations are expressed as and guaranteed by Service Level Agreements. We will present the core technology, discuss various security aspects and how they are addressed in the SmartLM prototype, and present the evaluation of the prototype through a number of usage scenarios. Finally, we will give an outlook on specific issues and current work extending the solution to Clouds and service based systems in general.

[4]By permission of Elsevier B.V

Future Generation Computer Systems 28 (2012) 1340–1349

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

SLA-based management of software licenses as web service resources in distributed computing infrastructures

Claudio Cacciari [a], Daniel Mallmann [b], Csilla Zsigri [c], Francesco D'Andria [d], Björn Hagemeier [b], Angela Rumpl [e], Wolfgang Ziegler [e,*], Josep Martrat [d]

[a] CINECA, 40033 Casalecchio di Reno, Italy
[b] Forschungszentrum Jülich GmbH, 52428 Jülich, Germany
[c] The 451 Group, London, WC1E6HH, United Kingdom
[d] Atos Origin - Research and Innovation, Barcelona, 08011, Spain
[e] Fraunhofer Institute SCAI, Department of Bioinformatics, 53754 Sankt Augustin, Germany

ARTICLE INFO

Article history:
Received 2 December 2010
Received in revised form
25 September 2011
Accepted 9 November 2011
Available online 18 November 2011

Keywords:
Distributed computing infrastructures
Service level agreements
Software licensing
Security

ABSTRACT

Until recently the use of applications requiring a software license for execution was quite limited in distributed environments. Due to the mandatory centralised control of license usage at application runtime, e.g. heartbeat control by the license server running at the home site of a user, traditional software licensing practices are not suitable especially when the distributed computing infrastructure stretches across administrative domains. In this paper we present a novel approach for managing software licenses as web service resources in distributed service oriented environments. Licenses become mobile objects, which may travel to the environment where required to authorise the execution of a license protected application. A first implementation has been realised for dynamic Grid environments in the European SmartLM project co-funded by the European Commission. The SmartLM solution decouples authorisation for license usage from authorisation for application execution. All authorisations are expressed as and guaranteed by Service Level Agreements. We will present the core technology, discuss various security aspects and how they are addressed in the SmartLM prototype, and present the evaluation of the prototype through a number of usage scenarios. Finally, we will give an outlook on specific issues and current work extending the solution to Clouds and service based systems in general.

1. Introduction

So far, commercial software is rarely used in Grids and public Clouds due to the limitations both with respect to the license management technology and the missing business models of the independent software vendors (ISV) for using their software in the Grid or the Cloud. Only in 2009 MathWorks has provided a technical solution (and a business model) allowing to use their MATLAB suite in the EGEE Grid environment [1]. However, this is a bilateral agreement only and has so far no implications for using MathWorks software in other Grids like the German D-Grid. Lately, IBM launched a cooperation with Amazon allowing IBM's customers to use own software licenses for a limited number of applications under certain conditions in the Amazon Elastic Compute Cloud (EC2), which is extending BYOSL from the IBM Cloud to a public Cloud. However, the use of the "bring your own software and license" (BYOSL) [2] option would have to be settled by IBM with each Cloud provider where a user wants to deploy and use an application.

The license management technology for software licenses is still based on the model of local computing centres providing both resources for computation and the software used for e.g. simulations together with the required licenses locally. Thus, these licenses are provided on the basis of named users, hostnames (IP-addresses), or sometimes as a site license for the administrative domain of an organisation. If we want to use this software in a distributed service oriented infrastructure, using resources that are spread across different administrative domains, that do not host the application's license server, we run into trouble. The licenses usually are bound to hardware within the domain of the user and access from outside to the organisation's license server usually is blocked, e.g. due to firewalls or missing authentication and authorisation mechanisms of the license server

* Corresponding author.
E-mail addresses: c.cacciari@cineca.it (C. Cacciari), d.mallmann@fz-juelich.de (D. Mallmann), csilla.zsigri@the451group.com (C. Zsigri), francesco.dandria@atosorigin.com (F. D'Andria), b.hagemeier@fz-juelich.de (B. Hagemeier), angela.rumpl@scai.fraunhofer.de (A. Rumpl), wolfgang.ziegler@scai.fraunhofer.de (W. Ziegler), josep.martrat@atosorigin.com (J. Martrat).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.11.005

suitable for a distributed computing infrastructure, thus, enforcing local use of the protected applications only. In contrast, Grid environments are usually spread across multiple organisations and administrative domains. Even worse, extending Grids to virtualised infrastructures, like utility and Cloud computing, introduces additional limitations since the underlying hardware and their performance indicators, e.g. CPU type and frequency are hidden, while they are often used in license agreements. Also, existing license management software lacks the possibility of reserving a license in advance for later usage. Instead, the user usually checks-out the license at job submission while the execution may be delayed for an indefinite time due to queuing systems and varying load especially when using remote Grid resources. The other approach, checking-out the license upon application start-up often fails, as the license has been checked-out by another user meanwhile and is no longer available. Thus, the user either blocks a license (being a scarce and often expensive resource) longer than it would be needed for running the application, or the user takes the risk that its application aborts because of an unavailable license when the application is launched after waiting (probably for a longer time) in a batch queue. There is no defined Quality of Service (QoS) for the user.

While current mechanisms almost inhibit the use of licensed software in Grid environments and virtualised infrastructures, the increasing role these environments play in resource provisioning requires a solution. Traditional licensing practices are under pressure from a variety of alternative options (software as a service, open source, low-cost development environments, and the increasing software piracy [3] etc.) and are tightening vendors' profit margins, pushing down licensing costs and giving more negotiating power to users. Licensors may expect licensees to buy additional licenses for each processor that executes the licensed software (multiplied software fees). This is definitely not viable in a Grid or Cloud context because it limits their flexibility and elasticity. On the one hand, software manufacturers need to change the way licensing works and use flexible and non-hardware based licensing solutions that better fit into a virtual environment (one of the top ten obstacles for Cloud Computing mentioned in [4]). On the other hand, Software as a Service is going mainstream as final users are asked to pay only for what they use. End users want fairness and flexibility and software vendors do not vote for a reduction in revenue. Hence, the achievement of a win win situation between software vendors and software users is the main requirement for a mutually advantageous change. The license management has to provide the technological basis for this. A license management product that overcomes current limitations and provides manifold benefits for all parties involved makes a compelling selling proposition here. The European SmartLM project [5] has provided a prototype for a generic and flexible licensing virtualisation technology based on standards for new service oriented business models.

To the best of our knowledge little research has been focusing on licensing technology since the new IT infrastructure paradigms – Grids, Clouds and SOA – became serious enhancements of traditional IT infrastructures. Early approaches like [6–8] propose front-ends to the FlexNet License manager [9] providing scheduling and reservation of licenses. However, both approaches assume open firewall ports at runtime to enable the communication between license manager and application. Dong et al. [10] focusses on maximisation of license usage and resource usage in Grids. Like the previously mentioned approaches, open firewall ports at runtime are a prerequisite. Other approaches like [11,12] stem from the P2P environment. The former addressing licensing of music sharing while the second one is more generally addressing content sharing. However, both approaches grant unlimited access or usage once a license has been issued and thus do not support a business model useful for ISVs. Katsaros et al. [13] finally is proposing a license mechanism suitable for SOA environments. However, the paper sketches the architecture and some possible interactions but lacks an implementation and experiments with real applications. Moreover, the approach also assumes open firewall ports at runtime. Only recently when these new paradigms gained ground in productive environments where e.g. more commercial simulation codes are used than in the e-Science domain license technology came to the fore. In [14] the authors give an overview on current licensing technology and models and describe two approaches developed in European projects to overcome the limitations. One of the presented approaches breaks with the current technology and has been implemented as a prototype in the SmartLM project while the second approach circumvents some of the limitations imposed by the de-facto standard of software licensing. The authors of [15] describe another approach, which is similar to SmartLM but lacks the integrated accounting and billing service and still requires network connectivity with the ISV when a user requests a license as the license management server is hosted by the ISV. As we will explain in the following sections, SmartLM has taken an holistic approach for all services around license management while effectively making dispensable the requirement for permanent network connectivity between the license management service, the execution site of the application during runtime or to the ISV when requesting a license. In the European project BEinGRID another approach was developed which allows the use of existing licenses in Grid environments through tunnelling of the communication of the license server to the application [16]. While technically feasible this approach raises a number of legal issues since many license contracts limit the use of a software license outside a company or outside a certain radius from the company.

The licensing and license management framework developed by the authors for the ElasticLM prototype aims to overcome the limitations described before by

- decoupling the authorisation for using a license from the authorisation for executing a license-protected application
- creating a secure software license token that my be deployed to the environment where it is needed for authorising the application execution
- making obsolete the communication current software licensing mechanisms require between license-protected application and the license server hosted by the user's home organisation at run-time
- providing defined QoS in terms of a Service Level Agreement as negotiated between user and license service regarding, e.g., availability of a certain license and corresponding features at a given time and for certain computing resources
- implementing sophisticated security mechanisms to secure the token and to allow the API to validate a token prior to forwarding the content to the application for authorisation.

The remainder of the paper is organised as follows. First, we give an overview of the approach taken for the SmartLM prototype and of the underlying business models. In Section 3 the architecture is presented. Section 4 discusses security aspects and Section 5 evaluates the prototype discussing use-cases that have been realised with the SmartLM prototype and presents times measured for the different steps of token creation. Finally, the paper concludes with a summary and plans for future research.

2. SmartLM overview

SmartLM is both the name of the European project and the name of the licensing and license management framework developed in this project. In this article SmartLM refers to the framework if not stated otherwise explicitly.

The SmartLM approach reflects the changing paradigms of the information technology. The transition from traditional monolithic infrastructures of computing centres to agile, service-based architectures was at the origin of the SmartLM project. Treating and implementing software licenses as Web Service resources, thus providing platform independent access to licenses just like other virtualised resources is the core of the SmartLM architecture [17]. In brief, SmartLM delivers the technology to provide License as a Service. Licenses are managed through a license service implemented as a bag of specialised services (see Section 3) and realised as tokens, which deliver the required flexibility and mobility. Decoupling authorisation for license usage from authorisation for application execution finally allows executing the license protected application without requiring a permanent bidirectional network connection to the license server at runtime controlling the authorisation for application execution, like it is the case today. Thus, the authorisation for using a license is done locally at the license service of the user's home organisation. This authorisation takes into account availability of the requested license and features, policies defined by the ISV (e.g. constraints regarding the nationality of a user for certain applications from ISVs with a headquarters in the US) and policies defined by the local administrator (e.g. usage times for production and test). All local authorisations for license usage are expressed and guaranteed by Service Level Agreements (see 3.2) based on the WS-Agreement [18] specification and result in a binding agreement between user and license service. The outcome of an authorisation is a digitally signed software token that contains all license-related information plus a number of certificates. This token is then deployed to the environment where the license-protected application is going to be executed. For deployment the orchestrator or standard mechanisms of the operating system are used, e.g. scp. The time for the deployment is thus depending on the network load and since the token has just a few kilobytes, the transfer time usually is significantly below 1 s. When the application starts-up the token is retrieved from the local file system, validated by the API and – if the token is valid – its contents delivered to the application to authorise the execution.

Moreover, this token-based solution allows license aggregation: an application service provider can provide the customer with access to applications without customer having to buy additional licenses. Customers' licenses may be combined with the ones owned by the service provider for running complex jobs with different applications. The license mechanism is designed for distributed environments and works with loosely coupled systems, realising software licenses as Web service resources renders licenses adaptable and mobile.

A built-in license scheduler allows licenses to be reserved in advance. With advance reservation a license can be guaranteed to be available at a later point in time, e.g. when the computing resources to execute the simulation become available. Thus, no risk of blocked licenses while an application is waiting in a batch queue for computing resources to become available. Similarly, there is no risk of aborted applications because the required license is used by another user at the time the application starts up after waiting for resources. An orchestration service can be used to synchronise license reservation and resource availability.

Using open standards as far as possible instead of proprietary protocols is considered crucial for the interoperability with and integration into existing middleware stacks. As mentioned before, in SmartLM license usage is governed by Service Level Agreements based on the WS-Agreement specification. A new term language has been defined allowing to express license properties as service description terms. Furthermore, in collaboration with the GRAAP working group [19] of the Open Grid Forum (OGF) the WS-Agreement specification has been extended to allow negotiation and re-negotiation (see Section 3.2). This joint work resulted in WS-Agreement Negotiation, the new specification that defines multi-step negotiation on top of WS-Agreement, which is subject to publication as OGF proposed recommendation in September 2011.

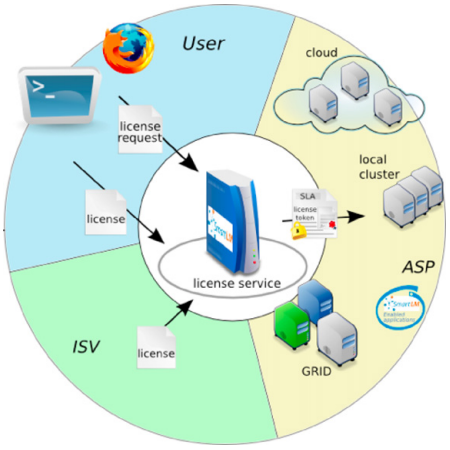


Fig. 1. SmartLM ecosystem.

The SmartLM project has identified three different actors: the user (and her home organisation, which is the licensee), the independent software vendor (ISV), which is the licensor and the application service provider (ASP), which provides the hardware and hosts the applications (see Fig. 1).

As mentioned before, SmartLM addresses the licensing management issues not only from the technological point of view, but also has developed new business models exploiting the capabilities of the prototype. The reason for looking into new business models is that for the ISVs, a business perspective using the new license technology must be present in order to convince them. The ISVs are not willing to adopt to new technologies when there is a risk of loosing revenue. Some of the new business models are discussed in the next section.

2.1. New business models

Through close collaboration with stakeholders – software vendors, application service providers and end users (all partners in the project) – we have identified major licensing shortcomings and have tried to fill in the gaps. The usage-based models presented in the following paragraphs look at different aspects of licensing problems.

1. In the first model we have pinpointed the *Application Service Provider* (ASP) and analysed five cases – customer license housing, embedded license (Dependant Software Vendor, DSV), license redirection (external consultant), license aggregation and license reselling – that companies may most frequently encounter in real operations. In all of them, the ASP plays a central role, being a reseller of hardware, software and services. The introduction of the ASP can be very advantageous for both the ISVs and the end users. From the *ISVs'* point of view, the ASPs can generate additional business offering licenses and hardware resources for end users on-demand (competitive resource provision). Making use of economies of scale and SmartLM features, the ASP can make existing models (e.g. short-term licenses) more attractive to customers, and introduce new ones that the ISV is not willing to offer, such as pay-per-use for license reselling.

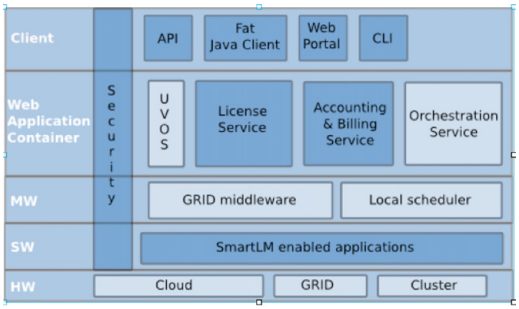


Fig. 2. SmartLM components.

2. The license extension model allows *end users* to extend their licenses in their Local Area Network (LAN) and distributed environments on-demand, e.g. for workload peaks. The license server takes care and simplifies the process of the extension of licenses, e.g. in terms of accounting and license administration. These mechanisms give end users more flexibility and value and at the same time generate additional revenue for ISVs and ASPs.
3. Currently, most contracts between ISVs and end users restrict the license usage to LAN. The license aggregation model allows the use of licenses that belong to different sites and brings them together to form a single license token. These licenses can come from either the ISV or the ASP. *End users* gain more flexibility and value and get access to huge hardware resources. The ASP provides these hardware resources to the end user and generates additional business for the *ISV*.

3. SmartLM architecture

The core part of the SmartLM prototype is the license service. Next to it there are the accounting and billing services, the orchestration service and the Unicore Virtual Organization System (UVOS) [20]. These last two components have been developed in different projects, but are integrated in the SmartLM architecture, as shown in the Fig. 2 where the components developed in SmartLM are the darker blocks. In particular, the use of UVOS (developed within the UNICORE project) does not imply dependencies, that would limit the usability of SmartLM to UNICORE environments. Rather, UVOS provides a middleware independent authentication, user attribute management and credential validation mechanism. As a consequence, the SmartLM prototype can and has been used for simulations with both UNICORE and Globus middleware stacks and without any middleware stack.

3.1. License service

The license service is capable to manage all licenses that support the new licensing mechanism owned by a company or organisation. It provides a single point for license management and an easy and comfortable access to the entire license information. For redundancy, multiple synchronised instances of the service may be hosted on different servers. Fig. 3 depicts the components of the license service and their interaction. When a company buys a license from a software vendor, the license is added to the license service using the *license administration service*. The *license management service* may then fork license tokens from this license once the user's request for a license has been accepted. Policies of the ISV and local policies are used by the *policy management service* to authorise the user's request. A user can request a license by means of one of the different clients available, such as the java based eclipse client, a web portal or the command line interface. Also, an application may follow the current practise and request a license at runtime. The authentication process always relies on the common UVOS service, allowing to store user attributes and validate the user credentials. When a user requests a license for an application or a feature of an application from the license service, the terms of license usage are negotiated between the user and the service through the *SLA and negotiation service* and then embedded in a Service Level Agreement document following the WS-Agreement specification. The negotiation is based on templates specific to the application. The cost resulting from license request is calculated beforehand and becomes part of the SLA. This allows to easily check the license cost against budget constraints for users or user groups implemented in UVOS. After creating the token the information relevant for accounting and billing is passed to the accounting and billing service through

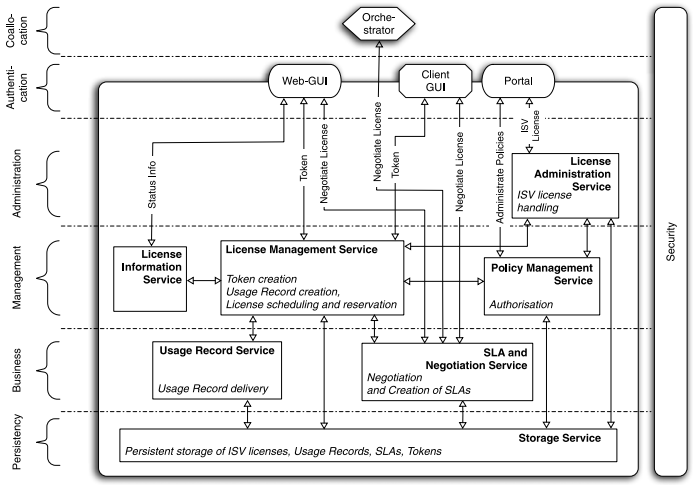


Fig. 3. Architecture of the license service.

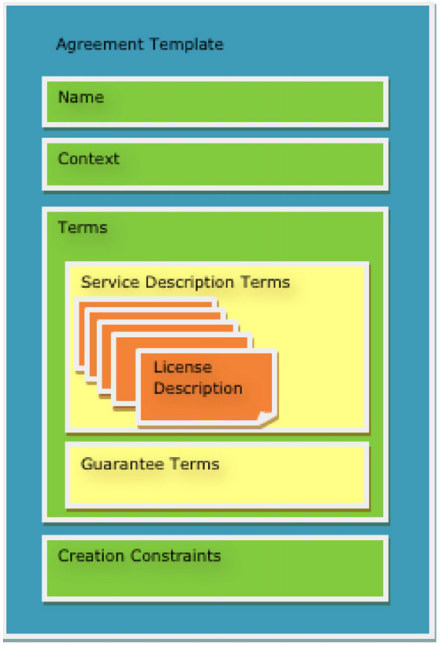


Fig. 4. SLA template for license agreements.

the *usage record service* in the form of a usage record. The *license information service* allows both users and administrators to retrieve information on installed ISV licenses, licenses available, reserved or in use. All persistent data of the license service are stored by the *storage service*. In addition there are a number of interfaces like the Web-GUI to access the *license information service*, the *license management service*, and the *SLA and negotiation service*, the portal to access the *license administration service* and the *policy management service*, and the Client API to access the *license management service* and the *SLA and negotiation service*. The *orchestrator* is an external component used to co-allocate licenses and features together with e.g. computing resources through negotiation with both license service and computing resources (through a Grid scheduler or a local queuing system). The result of the successful negotiation are SLAs and advance reservation for both licenses and computing resources.

In case of using remote resources the token will be forwarded to the site where the application will be executed either by the user, the *orchestrator* or the application. To protect against changes the token is signed by the license service, it may also be encrypted to prevent from unauthorised access when traveling towards the computational resource. Since the information about license and features could be used by competitors to estimate in which state of the development process for e.g. a new car a company is, the capability of encrypting the token is of importance for industrial companies.

3.2. Service level agreement and negotiation

The license service implements the WS-Agreement specification of the OGF for creating and negotiating Service Level Agreements (SLA). WS-Agreement specifies a language and a protocol for advertising the capabilities of service providers, for creating agreements based on templates, and for monitoring agreement compliance at runtime. See [21] for a discussion on WS-Agreement. Like an agreement document, the template is composed of a template name, a context element and agreement terms, but additionally also includes information on agreement creation constraints to describe a range of agreements which are acceptable by the agreement provider. Fig. 4 depicts a template with service description terms related to licenses.

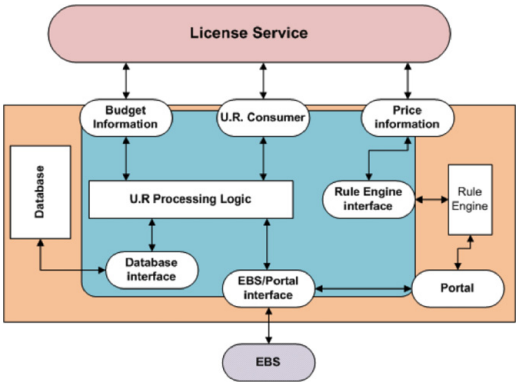


Fig. 5. Architecture of the accounting and billing service.

SmartLM has adopted and extended the WS-Agreement implementation for Java (WSAG4J) [22], which is a framework intended to provide an easy way to create, monitor and terminate service level agreements based on the WS-Agreement specification. According to the specification, agreements are created based on specific templates. Therefore, a client queries the agreement templates from a SLA factory service, which exposes them as resource properties. In ElasticLM templates are dynamically created based on the available licenses installed in the license service. The user looks up the available templates (e.g. through the client GUI or the orchestrator GUI) selects a suitable template (with license and features needed) and creates a new agreement offer, and in doing so may modify the template respecting the creation constraints. The offer is then sent to the factory service (part of the SLA and negotiation service) that will create a new agreement and return an Endpoint Reference (EPR) to it. WSAG4J framework provides also the capability of negotiating templates, which is an improvement of the initial WS-Agreement protocol. When the user changes the retrieved template, it may create a negotiation quote with the modified template and send it to the agreement factory service, which will try to find a suitable template based on the quote. If a template can be provided, the agreement factory service just returns it to the client, indicating that an offer based on that template will potentially be accepted, otherwise it employs some strategy to create reasonable counter offers.

3.3. Accounting and billing service

The architecture of the accounting and billing service is presented in Fig. 5. It consists of three main blocks: the database to store the usage records, the usage record processing logic for processing the usage records and the rule engine to determine the price of a license. Finally, the accounting and billing portal is used by different SmartLM users to access to the full accounting and billing information generated by licensed applications. In addition, it provides functionalities like budget control triggers of the licensed software, and to define different business rules used to define different pricing policies.

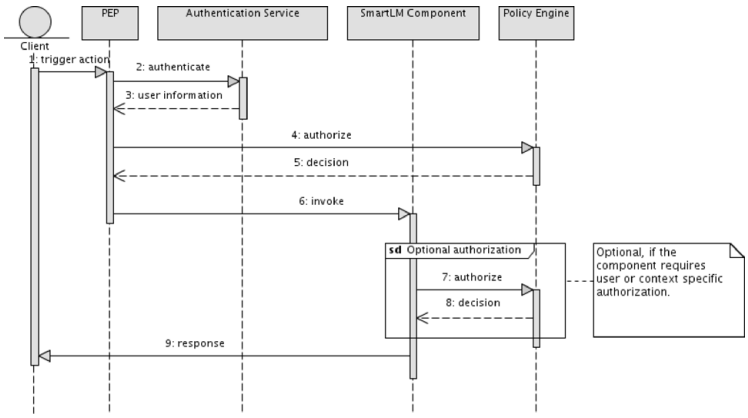


Fig. 6. Authentication and authorisation sequence diagram.

4. Security

All the positive aspects considered so far about the flexibility and the reliability of the SmartLM services would be useless without an adequate level of security. The security issues can be examined from two points of view: the services and the licenses.

4.1. Services

Service security in SmartLM is not much different from service security in other distributed, service based, systems. In the same way as the most important web service engines like Axis2, CXF, etc., SmartLM employs a handler scheme, where incoming and outgoing calls pass through handlers, which deal with authentication and authorisation in a generic fashion. For incoming calls that means that only authenticated and authorised calls will even make it to the service implementation itself. Unauthenticated or unauthorised calls will be rejected already by the incoming security handlers. The prototype follows this scheme and implements SmartLM specific handlers for authentication and authorisation. Fig. 6 shows a generic sequence diagram of a SmartLM component being accessed and doing authentication and authorisation. A client – a user or another SmartLM service – triggers an action in a SmartLM component. This component needs to first authenticate the client and then determine its permissions, i.e. if it is allowed to trigger the requested action. Therefore, it will first call the authentication service (UVOS) to query information about the client. With this information, it will query the policy engine (XACMLight) [23], whether the client is authorised for this action or not. The policy engine will return an appropriate reply, permitting or denying access. The use of UVOS allows us to use a single user database for both middleware and the license server wherever these are located in the same administrative domain.

Policy enforcement is done before any request enters the service. This allows the use of generic authentication and authorisation components, which can be enabled or disabled for services as needed. Besides authentication and authorisation do not have to be hard wired into the services themselves. As you can also see from Fig. 6, we allow for an optional authorisation step inside the service. While this is not generally necessary, as most of the authorisation can be done generically, we foresee an occasional need for service specific authorisation, which will be allowed through this. This is true in cases when the call to a service interface doesn't allow proper distinction of what resource is affected by the actual call. The service itself may however be able to make this distinction.

4.2. Licenses

Licenses and license tokens are the primary goods in SmartLM. They need to be protected against unlawful use by all means. We expect that all parties involved in the SmartLM license scenario have X.509 certificates [24], which enables them to sign and verify the information transported in a certificate. This expectation cannot be satisfied under all circumstances. Most of the time it will be sufficient that only ISVs and license servers at ASPs have full licenses. They are required for the verification of the chain of trust from the ISV to the license server. User certificates are only really needed in Grid scenarios, where the job submission needs to be signed, such that the user cannot refute to have sent the job when billed for its execution. When a service provider buys a license from the ISV, that license will be issued exactly to this service provider's license server. The license server's identity will be part of the license. The entire information will be signed, so it is not possible to copy the license to another license server without also copying the server's identity, i.e. its private key, to the other server. Optionally, in order to avoid the lodging of the license file in other servers, certain properties of the license server's hardware can be included in the license servers identity. Checking out a license token is similar to buying a license from an ISV (see Fig. 7). Again, the token will be signed including information about the receiver of the token, e.g. the job. In order to protect the token against unlawful use, hashes of the job's input, which have been provided by the user on submission, are contained in the token. Thus, the token is only useful for a particular execution of the software. Different input would require a new token. The license server has to ensure that the sum of the multiplicities of a feature in all license tokens derived from a particular license may never exceed the multiplicity of the respective feature in the parent license. For example, if a license contains permissions to use a feature on 100 CPUs at any given point in time, then the sum of CPUs for this feature in all the license tokens derived from this license may not exceed this number at any time. This requirement has to be ensured by the license server. This is actually one of the key features where the ISV has to trust the license server to observe this requirement. This trust is expressed by issuing a license to this server and can be verified by the application. The purchase and checkout procedure can be summarised in the following scenario. A company buys a license from an ISV. For this purpose, the company provides its license server's public certificate to the ISV. The ISV then issues the license to the license server of the company. This is ensured by attaching the server's credential information to the license.

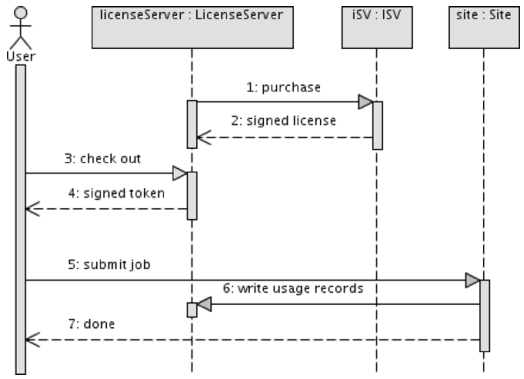


Fig. 7. Purchase and checkout sequence diagram.

It can thus be verified that this license is issued specifically to this license server. Only the license server to which a license has been issued can fork license tokens from this license. Information about the original license needs to be conveyed inside the license token for verification. At the same time, the license token has to contain information about whom it has been issued to and who is allowed to use it. Eventually, the Policy Decision Point (PDP) in the application will be able to verify if a correct path of delegations (ISV to License Server to User) has been built and obeyed for the execution. The purchase and checkout procedure is depicted in the sequence diagram presented in Fig. 7.

5. Evaluation of the SmartLM prototype

For an evaluation of the results we present and discuss different license usage scenarios that have successfully been realised during the evaluation of the prototype. For the evaluation we used three geographically distributed Grid testbeds with Globus toolkit and UNICORE as middleware stacks. The orchestrator was only used in a UNICORE environment to co-allocate licenses and computing resources because the Globus toolkit did not support advance reservation through its job submission interface. Additionally, we present times measured for the different steps of token creation.

The SmartLM architecture allows for different usage scenarios of license protected applications. The main distinction of the different scenarios is the availability of a network connection between the SmartLM license server and the protected application at runtime.Within the SmartLM project three ISVs adapted their applications for using the SmartLM API that enables the new license mechanism: with CFX from ANSYS [25], PERMAS from INTES [26], and OPTIMUS from LMS [27] three heavily used industrial codes are included in the project's use-cases and evaluation. Naturally, replacing the API of a traditional license management system against the SmartLM API is a prerequisite for using the new mechanism.

5.1. Basic scenario: without network link available at runtime

Fig. 8 presents the basic SmartLM license usage scenario which covers cases where no network connection to the SmartLM license server is available during the execution of the protected software. The following two use cases have been used to validate this scenario:

(a) a compute cluster that executes the application is protected through a firewall that does not allow any communication from the cluster to the outside world and vice versa,

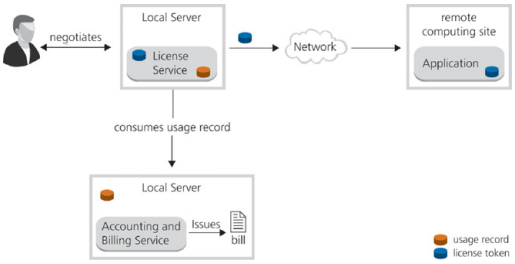


Fig. 8. Basic scenario without bi-directional network connection at runtime.

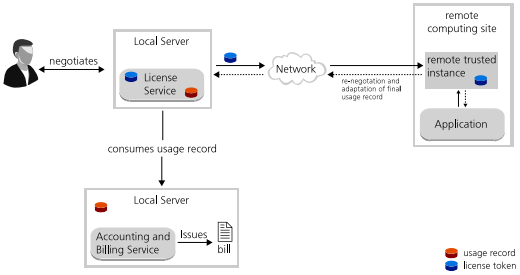


Fig. 9. Advanced scenario with bi-directional network connection at runtime.

(b) the application is used on a laptop while traveling without network connection at all.

For both use-cases the following steps were performed

(1) negotiate the usage of the license protected software in advance, reserve the license enabling other users using the license before and after the reservation period while preventing other user from blocking the license during the reservation period. The SLA the user creates when creating the token guarantees the availability of the license while the license scheduler creates the schedule of license usage
(2) receive a license token from the SmartLM license server and
(3) transfer the license token to the execution environment before the application is executed.

In a scenario without network connection a modification of the license terms at run time–time is not possible, i.e. the user is charged as agreed in the SLA even when the program uses less time than expected or is aborted.

5.2. Advanced scenario: network link available at runtime

The advanced SmartLM license usage scenario depicted in Fig. 9 covers all use cases where a network connection between the protected application and the SmartLM license server is available during execution, either through a trusted entity or a direct connection from the API of the application to the license server.

Two use-cases have been used to validate the advanced scenario:

(a) a user submits a job for batch execution to a Grid scheduler that chooses the executing resource by a user defined policy, e.g. optimisation function of time and cost
(b) a user starts an interactive application on a workstation and the token is requested on the fly.

Table 1
Overall temporal behaviour of the token generation.

Component	Time (ms)
Scheduler and HQSLDB (in memory DB)	215
License Management Service (LMS)	160
SLA and negotiation service (including LMS-adapter, WSAG4J, HQSLDB)	1900

Table 2
Temporal behaviour of the SLA creation.

Operation	Time (ms)
Get template	12
Creation of the License Management Service (LMS) Adapter	70
License reservation	370
Lookup of requested license/feature combination in scheduler DB	720

In a scenario with a Grid scheduler, it will schedule the license token along with the application or the data required for the application execution. For that purpose SmartLM is shipped with an external Orchestrator which provides the Grid scheduler functionality. In the advanced scenario re-negotiation of license terms is possible, i.e. extending or reducing the terms of a license like reservation time or features. This allows to return a license to the license scheduler when the application stops earlier than predicted by the user, e.g. because of a failure. However, local policies may be used to control if tokens can be returned and if there are penalties or rewards defined for this case. The policies will be expressed in the terms of the SLA. The token contains the EPR of the agreement. This EPR allows the API (or the trusted instance) to start the re-negotiation process of the proper agreement.

5.3. License aggregation

The token-based licenses leverage new forms of license aggregation for both scenarios described before. A user having access to local licenses is able to aggregate them with external licenses owned e.g. by an ASP when using the resources of his ASP to satisfy peak demand. Aggregation is a simple as moving the token generated from the local license to the ASP environment, creating a token from the ASP's license and providing both to the SmartLM API for authorisation. The API then aggregates the license terms from two (or more) tokens and communicates the result to the application. For the evaluation we used the ANSYS CFX application to make use of multiple tokens from different sources to make all features available that were required for a more complex simulation.

5.4. Required times for the different steps of token creation

In order to validate that the response time of the system is suitable for a productive environment we measured the temporal behaviour of the prototype when creating a token. Table 1 depicts the average distribution of time spent in different components during token creation. As can be seen the major part (1.9 s) of the total time of about 2.3 s is being used for creating the agreement.

We did a further analysis of the behaviour of the SLA and negotiation service and measured the time for different operations during the createAgreement step in more detail. The result is shown in Table 2. The analysis indicates, that half of the time (about 0.7 s) is used for browsing through the license scheduler's database looking for the appropriate combination of license and features as requested by the user. Clearly, this shows the greatest room for optimisation. Another optimisation approach would be integrating the SLA and negotiation service with the license management service, which would reduce the 0.07 s spent in the adapter (as this would become obsolete) and roughly 0.15 s of the time required for license reservation.

6. Conclusion and future work

We discussed a new approach for software licensing in distributed computing infrastructures based on Service Level Agreements. We presented the basics of the SmartLM technology and implementation, new business models considering the interest of the software vendors, the users and the application service providers. Managing software licenses as Web service resources allows to achieve the flexibility required by distributed environments such as Grid and Clouds, overcoming the limitations of the actual licensing technologies. The presentation of the technology is concluded by a section presenting the results of the evaluation of the prototype. We presented the major use-cases for the evaluation and discussed the temporal behaviour of token generation.

At the end of the SmartLM project three partners of the SmartLM consortium (Fraunhofer SCAI, ATOS Origin, and Gridcore AB) have agreed on a partnership for transforming the SmartLM prototype into the commercial product for software licensing elasticLM [28]. The first release for early adopters has been created by the end of 2010, while an updated release has been available since Summer 2011. Besides stability the development focusses on additional security and trustability of the license tokens, especially in cases when an application does not need input data. Currently, the second release is undergoing more intensive stress tests to evaluate the behaviour of the system with respect to response time and stability when a large number of tokens is requested during a period of a couple of weeks. Moreover, additional enhancements are under development in the recently started European project OPTIMIS [29] like (i) the introduction of trusted clocks limiting the possibilities of cheating by manipulating time and date of the execution environment (is part of the API since spring 2011), (ii) the realisation of a trusted entity deployed in the Cloud and able to manage multiple tokens created beforehand for the execution of one or multiple applications, and (iii) the deployment of a copy of a site's license service into the Cloud with the capability of issuing tokens from a subset of licenses owned by this site (while reducing the set of licenses and features at the site by the same amount).

Acknowledgments

Some of the work reported in this paper has been funded by the European Commissions ICT programme in the FP7 project SmartLM under grant #216759 and is currently funded under grant #257115 in the FP7 project OPTIMIS. This paper also includes work funded by the German Federal Ministry of Education and Research through the D-Grid project under grant #01AK800A. Last but not least the authors gracefully acknowledge the stimulating discussions in the GRAAP-WG.

References

[1] The MathWorks enables MATLAB parallel computing tools to run on the EGEE grid, 25 September 2011. Web site: http://www.mathworks.de/company/pressroom/The-MathWorks-Enables-MATLAB-Parallel-Computing-Tools-to-Run-on-the-EGEE-Grid.html.

[2] IBM Licensing for Amazon Cloud web site, 28 November 2010. Web site: http://www-01.ibm.com/software/lotus/passportadvantage/pvu_for_Amazon_Elastic_compute_cloud.html.

[3] Seventh annual BSA/IDC global software-09 piracy study, 25 May 2010. Web site: http://portal.bsa.org/globalpiracy2009/index.html.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: a Berkeley view of cloud computing, Technical Report UCB/EECS-2009-28, University of California, Berkeley, 2009.

[5] SmartLM—grid-friendly software licensing for location independent application execution, 18 July 2008. Web site: http://www.smartlm.eu/.

[6] Xiaoshe Dong, Yinfeng Wang, Fang Zheng, Hua Guo, Shuncheng Yang, Weiguo Wu, Floating license sharing system in grid environment, in: SKG, IEEE Computer Society, 2005, p. 96.

[7] Wei Fu, Nong Xiao, Xicheng Lu, Sharing software resources with floating license in grid environment, in: NPC'07: Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops, IEEE Computer Society, Washington, DC, USA, 2007, pp. 288–294.

[8] Feng Guofu, Wang Yinfeng, Guo Hua, Dong Xiaoshe, Research on software license manager and sharing system in grid, in: GCCW'06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops, IEEE Computer Society, Washington, DC, USA, 2006, pp. 35–38.

[9] Flexera Web site: http://www.flexerasoftware.com/products/flexnet-manager.htm.

[10] Xiaoshe Dong, Yinfeng Wang, Fang Zheng, Zhongsheng Qin, Hua Guo, Guofu Feng, Key techniques of software sharing for on demand service-oriented computing, in: Yeh-Ching Chung, José E. Moreira (Eds.), GPC, in: Lecture Notes in Computer Science, vol. 3947, Springer, 2006, pp. 557–566.

[11] Sai Ho Kwok, Siu Man Lui, A license management model for peer-to-peer music sharing, International Journal of Information Technology and Decision Making 1 (3) (2002) 541–558.

[12] Yang Liu, Chun Yuan, Yuzhuo Zhong, Implementing digital right management in p2p content sharing system, in ICA3PP, 2007, pp. 348–355.

[13] Gregory Katsaros, Savvas Antonopoulos, Dimosthenis Kyriazis, Theodora Varvarigou, Service oriented license providing, in: Proceedings of IEEE International Conference on Service-Oriented Computing and Applications, SOCA, 2009.

[14] Jiadao Li, Oliver Wäldrich, Wolfgang Ziegler, Towards SLA-based software licenses, 2008, pp. 139–152.

[15] M. Dalheimer, F.-J. Pfreundt, GenLM: license management for grid and cloud computing environments, in: Proceedings of the CCGrid Conference 2009, 2009.

[16] Yona Raekow, Christian Simmendinger, Ottmar Krämer-Fuhrmann, License management in grid and high performance computing, Computer Science - Research and Development 23 (3–4) (2009) 275–281.

[17] Björn Hagemeier, Daniel Mallmann, Wolfgang Ziegler, Securing software licenses in a location independent manner, in: Proceedings of Cracow Grid Workshop 2009, 2010, pp. 112–119.

[18] A. Andrieux, et al. Web services agreement specification (WS-agreement), Grid Forum Document GFD.107, Open Grid Forum, 2007.

[19] Grid Resource Allocation Agreement Protocol Working Group, 2008, 18 July 2008. Web site: https://forge.gridforum.org/projects/graap-wg/.

[20] UVOS—Unicore Virtual Organisation System, 8 February 2010. Web site: http://uvos.chemomentum.org/.

[21] Dominic Battré, Matthias Hovestadt, Oliver Wäldrich, Lessons learned from implementing WS-Agreement, in: Philipp Wieder, Ramin Yahyapour, Wolfgang Ziegler (Eds.), Grids and Service-Oriented Architectures for Service Level Agreements, Springer, US, 2010, pp. 23–34.

[22] WSAG4J—web services agreement for Java, 8 February 2010. Web site: http://packcs-e0.scai.fraunhofer.de/mss-project/wsag4j/index.html.

[23] Xacmllight sourceforge project web site, 25 September 2011. Web site: http://sourceforge.net/projects/xacmllight/.

[24] IETF X.509 specification, 25 September 2011. Web site: http://www.ietf.org/rfc/rfc3280.txt.

[25] ANSYS Germany web site, 25 September 2011. Web site: http://www.cfx-germany.com/.

[26] INTES Engineering Software web site, 25 September 2011. Web site: http://www.intes.de/.

[27] LMS International web site, 25 September 2011. Web site: http://www.lmsintl.com/Belgium.

[28] IelasticLM product web site, 25 September 2011. Web site: http://www.elasticlm.com.

[29] OPTIMIS project web site, 25 September 2011. Web site: http://www.optimis-project.eu/.

Claudio Cacciari got a Degree in Physics in 2001 at the University of Bologna (Italy). He has been working since 2002 at CINECA, the biggest Italian supercomputing center, in the High Performance Computing area and since 2004 he has been involved in European projects. He has experience in software integration, Service Oriented Architecture (SOA) tools, Grid and Cloud environments. He is collaborating to some important FP7 European infrastructural projects, such as PRACE for the computing and EUDAT for the data management, and to EMI, a project which fosters the convergence among different European Grid middleware tools. In particular, he is leading the EUDAT task related to the technology appraisal, which implies the evaluation of the available technologies according to the user requirements. He was one of the developers of the core services of the SmartLM project.

Daniel Mallmann studied Electrical Engineering and received his diploma from the RWTH Aachen in 1997. He works in the Jülich Supercomputing Centre as a research scientist on distributed systems and since 2010 as head of the division "Federated Systems and Data". The division is the major development partner of the European Grid middleware UNICORE and involved in national, European, and international projects including PRACE, EGI, EMI, and XSEDE. Daniel was involved in several Grid projects since 2000 and coordinated the FP6 project UniGrids.

Csilla Zsigri is the Director of Consulting Services for EMEA, at The 451 Group, a technology-industry analyst and data company. Previously, Csilla worked as a business consultant for Atos Origin's Research and Innovation unit in Barcelona, where she led business and marketing activities in numerous European ICT Research and Technology Development projects. She led the work of these organizations focusing on the analysis of target markets and competitive landscape, generation of business models and the elaboration of go-to-market plans. Before joining Atos Origin, Csilla worked for a Hungarian management consulting company. Csilla holds an Economist MSc degree specialized in Business and Management Consulting.

Francesco D'Andria is a project manager of IT research initiatives at ATOS. He has been involved in several European and National research initiatives that have contributed significantly to the development of his professional expertise. He has research experience in the scientific areas of Computational Grid, High Performance middleware for distributed systems, Cloud Computing, SOA and SLA. He is interested in the employment of these technologies in the business world. He has also investigated the use of Commodity Technologies in order to implement Grid Services and their integration with COTS components. His research activity on these topics started in 2000 during his thesis period, with the title: "SIMULAZIONE IN H.L.A. E POSSIBILE INTEGRAZIONE IN.NET". Actually Francesco is coordinating the EU FP7 STREP Project Cloud4SOA.

Björn Hagemeier received his diploma in Computer Science from the University of Paderborn, Germany in 2006. He is currently employed as a researcher at the Jülich Supercomputing Centre (JSC) of Research Centre Jülich (FZJ). He is a developer of the UNICORE Grid middleware, predominantly active in the development of clients. He has a strong interest in virtualization and cloud computing matters. Prior to the work at FZJ, he gained experience as a developer of a mesh generation and modeling application for FEM methods. He has also been working as an IT consultant in several projects and as a project manager with a software license company.

Angela Rumpl works on a broad range of subjects for both national and European research projects including Grid and Cloud technologies, software license management as well as bioinformatics. Angela's first professional post was to work as a scientific software developer for the very successful European project SmartLM. After three years Angela moved on to coach her team as a Scrum Master to become more efficient in creating highly productive software products. Angela continues to progress her passion for agile software development and her enthusiasm to getting products well done.

Wolfgang Ziegler is head of the Grid Middleware Research Group of the Bioinformatics department of SCAI with more than 20 years experience in distributed computing and a long record in Grid R&D. He has been active in the US Grid Forum and later the Global Grid Forum since 1999. He was a co-chair of the Global Grid Forum (GGF) GRID Scheduling Dictionary Working Group and he is a co-chair of the Open Grid Forum Grid Resource Allocation Agreement Protocol Working Group responsible for the WS-Agreement specification. He had a leading position in a number of European projects, e.g. in the executive committee of the CoreGRID Network of Excellence, in the Technical Board of the PHOSPHORUS project. He was the scientific co-ordinator of the SmartLM project. He is co-author of more than 80 conference papers and journal articles.

Josep Martrat Degree in Telecommunication Engineering, Polytechnic University of Catalonia (1999). He is currently Market Manager of Telecom, Media and IT services at Atos Research & Innovation Department. His research activities concern among others the study of Cloud technologies, the use of SLAs and their application in business world and new services in Future Internet. He has long experience in large software R&D projects and he is currently Project Coordinator for BonFIRE project about building an multi-cloud experimental facility. He is also collaborating in IP OPTIMIS project about optimizing services deployment on hybrid cloud computing considering parameters of trust, risk, eco-efficiency and cost. In addition, he previously coordinated SMARTLM (STREP FP7) on license management in distributed computing. He also collaborated in AKOGRIMO (IP FP6). He has experience in real-time software design and development in international projects in communications domain.

Paper V

Expressing Quality of Service and Protection Using Federation-Level Service Level Agreement[5]

L. Blasi[1], J. Jensen[2], and W. Ziegler[3]

[1]Hewlett-Packard, Italy
[2]STFC, UK
[3]Fraunhofer, Germany

Abstract: Frameworks for service level agreements (SLAs) have been developed to allow services to discover and negotiate SLAs dynamically, without direct human intervention. In this brief paper, we give a description of the experiences of two projects which are building on existing work in SLAs: the main advantages being to obtain better overall pricing of services for the consumer, and the SLAs are useful as a building block or extension of cloud federations. We also argue that the "Quality of Protection" for some types of clients is as important as the Quality of Service.

[5]By permission of Springer Verlag

Expressing Quality of Service and Protection Using Federation-Level Service Level Agreement

Lorenzo Blasi[1], Jens Jensen[2], and Wolfgang Ziegler[3]

[1] Hewlett-Packard, Italy
[2] STFC, UK
[3] Fraunhofer, Germany

Abstract. Frameworks for service level agreements (SLAs) have been developed to allow services to discover and negotiate SLAs dynamically, without direct human intervention. We give a description of the experiences of two projects which are building on existing work in SLAs: the main advantages being to obtain better overall services (including pricing) for the consumer, and the SLAs are useful as a component of, or extension to, cloud federations. We also argue that the "Quality of Protection" is an important part of SLAs.

1 Background

Traditionally, Service Level Agreements (SLAs) are negotiated between service providers and the customer. The SLA typically describes the terms and nature of the service, as well as specific parameters of the service, perhaps including the cost, target uptime (if it's an online service), level of support, etc. The aim is to have each party agree on the type and level of service provided. Traditionally, SLAs are negotiated between *people* who represent the provider and consumer – in particular, they must have the *authority* to negotiate an agreement.

The availability of online services, and cloud services in particular, changed this scenario. Now the consumer is a person looking for a service, and the provider publishes terms and conditions (T&C), cost, target availability on a web page; and users may need to accept T&C prior to using the service. For a typical cloud service, the customer's needs are small compared to the provider's capacity, and there is no need to enter into negotiations, nor would the provider be interested in negotiating with a large number of small users. When "shopping around" for services, the user can either "take it or leave it'.' For a cloud service, the provider is an automated agent which can deal fully with setting up the new customer. All customers are then treated the same, leading to an economy of scale.

The logical extension is: what if both the customer and the provider are represented by agents? And what if they can not just sign an agreement, but also negotiate terms? In this extended model the human outlines a policy for service delivery by expressing requirements, and leaves it to the agent to look for service providers that match.

The promise of this approach is obvious: now the end user is free to consume services, perhaps simultaneously from multiple providers (assuming of course

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 146–156, 2014.

they are interchangeable). If a provider drops out, or another becomes cheaper, the agent should negotiate new agreements, and services will eventually move to the new provider.

Moreover, brokering and monitoring services from multiple providers can be a means of obtaining higher Quality of Service (QoS), at least availability and perhaps bandwidth, as in the "InterCloud" vision [5]. This approach could, however, reduce the protection of confidential data, as data may have to be replicated to multiple providers, and an agent may need to have the rights to move it without the owner's intervention. In such a case, a Quality of Protection should be part of the SLA.

The work presented here is the joint experience of two EU-funded projects. Contrail [4] is reusing an SLA framework from a third (earlier) project called SLA@SOI [3]; the OPTIMIS SLA [2] is based on the WSAG4J framework. Contrail is building IaaS federations which can manage cloud resources from multiple providers. OPTIMIS is managing IaaS provider attributes, to orchestrate applications (which may be legacy applications) on the cloud, specifically aiming to manage trust, risk, eco-friendliness, and cost as service parameters.

2 Existing Work

Grid information services are based on LDAP: originally Globus MDS [21], they have evolved into schemata known as GLUE [19] which can be expressed not just in LDAP but also in other formats (e.g., JSON or XML). Grid information services publish the state and availability of the service at some (hopefully recent) point in time, and dynamic data is refreshed regularly; they can be used as non-guaranteed service discovery as well as accounting.

Ruiz-Alvarez and Humphrey [1] developed a framework for describing cloud storage providers: the published information allows a customer to select services based on attributes. Like the grid, it is an instantaneous snapshot of the attributes, valid at the time it is created, and, as cloud service providers do not generally publish this information, requires some other party to update the information from time to time. Formatted in XML, the document includes a description of the providers' interfaces, security, geographic location, as well as samples of past measurements of performance metrics.

Indeed, once we move into negotiated cloud services, a number of requirements and failure modes can be recognised [20], Sec. 3.9, some of which are familiar from distributed computing (transient errors, credential timeouts), some of which could arise in heterogeneous distributed environments (inconsistent naming of capabilities, interoperation problems), and some may be specific to the cloud (the offer will need to be timely and accurate.)

Service discovery and matchmaking are well studied concepts in distributed systems, and a full survey of this work is beyond the scope and the focus of this paper. However, some of the experiences in matchmaking are equally valid for the present work. For example, [9] suggested the need for well-defined semantics, a common possibly extensible framework for agreement/negotiation, the need

to accommodate heterogeneity in services, the need for sufficiently expressive queries/matching, and perhaps allowance for a certain "fuzziness" in matching. Like the much earlier [12], the use of semantic web was proposed.

In the Italian Cloud@Home research project [14], the resource is opened not to a specific project but to a possibly unknown customer. More recently, the project looked at the use of SLAs [15]: the QoS for negotiation is based on monitored availability and estimated performance. Potentially, the provider could be an "at home" user (as with BOINC [16]), but the project currently only looks at spare resources in private clouds.

The "InterCloud" project [5] further investigated SLAs for QoS, but mostly aiming for execution time in PaaS [8]. This work relied on an estimation of the execution time of individual "jobs" along with an overall view of the expected number of jobs.

Service Oriented Architecture (SOA) models may look at QoS for web services (e.g., WS-ReliableMessaging) [17], Chap. 13. If we are orchestrating complex services from multiple cloud providers, the lessons of providing QoS with reliable web services becomes relevant. This orchestration of services is, however, at a higher level compared to the approach we take in this paper, and could usefully be the subject of future investigations. More generally, there has been work on federation of clouds, some of which will be touched upon in this paper. NIST is very active in relevant cloud activities [18] and OGF, DMTF, and SNIA are working on standards to enable federations.

3 Use of Standards

WS-Agreement [13] and WS-Agreement Negotiation [11] are open standards from the OGF [10]. To define or contribute to a new standard, projects need to start early, so the work on the standard has a chance of completing during the lifetime of the project. The use of open standards bodies is encouraged so other parties have a chance to contribute, or help test interoperability. While using a standard is no guarantee for interoperability, it reduces the complexity and effort to get two code-independent implementations of, e.g. a SLA negotiation, to successfully cooperate.

4 Federation and SLAs

4.1 Federation Model in Contrail

A Federation in Contrail is a software infrastructure managing a set of independent IaaS providers (but could also encompass storage as a service). Providers are typically competing private organisations, with their services differentiated along the three major axes of price, Quality of Service (QoS) and Quality of Protection (QoP).

Contrail services can be applied also when providers collaborate, such as community clouds, or when several datacentres are connected to the same federation to form a big private cloud. The Contrail federation can thus also be an enabler for Enterprise-level private clouds.

The software infrastructure offered by Contrail is structured into three main levels: *federation*, *provider* and *resource*. Briefly, the resource is the actual IaaS (or other resource), the provider is an abstraction with a common interface which publishes services and serves as an endpoint for SLA negotiations, and the federation provides the overarching control, aiming to realise the "Multi-Cloud" or "Cloud of Clouds" visions [5], [6]. In SLA Management at federation level, user requirements are expressed in a uniform way, independently of the underlying resources, thus enabling scenarios such as Cloud Brokering, Cloud Aggregation and cross-Cloud SLA Enforcement.

Once requirements are defined, the federation looks at available providers and negotiates with them. Actual resources (such as IaaS) are then provided at the resource level (so the provider must know how to authenticate to the resource on behalf of the user), and whenever possible, the provider level monitors the service provided by the resource.

4.2 OVF and SLA for Describing Complex Infrastructures

In Contrail, users can define an "*application*," an infrastructure configuration made up of one or more layers of Virtual Machines (VMs) connected by virtual networks. Applications use OVF, an open standard, to define the resources required for the application. An SLA then refers to a specific OVF descriptor file and expresses guarantees about specific OVF items, such as Virtual Systems. Each OVF VirtualSystem in Contrail specifies a VirtualMachine type (or template) of which multiple instances can be created at provisioning time. The same SLA document can express different guarantees for different items described in the OVF. For example the same SLA can specify that instances of VirtualSystem A must be located in UK and instances of VirtualSystem B must be located in France, thus making it possible to build a disaster recovery system based on two distinct geographical sites.

4.3 Automatic SLA Negotiation and Dynamic Pricing

When the user has described their application using OVF and identified in the SLA proposal all the guarantees which are needed on each element, they can initiate a negotiation via the Contrail Federation. The system will propose a price for each part of the application and will let the user experiment by negotiating different configurations until the right balance between price and QoS is found (Fig. 1).

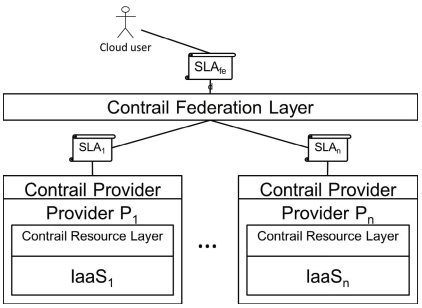


Fig. 1. Multi-level SLA negotiation

The provider layer automatically generates an SLA offer complete with a price for the resources described by the OVF and the QoS in the user's SLA proposal. The price for each VirtualSystem defined in the OVF depends not only on the amount of resources (e.g., memory or number of cores), but also on all the guarantees applied to it and identified in the SLA offer (e.g., location or co-location). The price is automatically calculated by the system for the required SLA+OVF configuration on the basis of the standard price list of that provider, but it could also be customised for the specific user by applying personalised marketing techniques. This personalisation helps the provider attract more customers, and weighing price/QoS helps users find the best value for money.

When a single provider cannot satisfy all the requirements, the federation can split the application and deploy it as an ensemble of cooperating parts on different providers. Splitting the application will go along with splitting the user's SLA proposal: the federation will negotiate each SLA part with different providers, and at each negotiation round will transparently provide to the user a SLA offer aggregating the best SLA offer for each of the parts. More on SLA splitting can be found in [7]. Further scenarios involving application and SLA splitting include scaling across multiple providers, Cloud bursting, and cross-provider SLA violation enforcement.

4.4 Proposed SLA Terms

Table 1 describes the QoS and QoP terms used (or planned) in Contrail SLAs.

QoS terms for IaaS providers can either express guarantees on the whole infrastructure, such as *availability*, or guarantees on a single resource, such as the *cpu_speed* of a specific VM. Most of the terms supported by Contrail are on single resources. When comparing SLA offers from different providers to find the best one, the value of some terms can be maximised (e.g. *storage_reliability*) or minimised (e.g. *price*), while the value of other terms must match exactly the user request in the SLA proposal. One such term expresses the geographical *location* of computing (VMs) or storage resources: in this case a SLA offer to be selected must match exactly the required countries

The term *Not_co_locate_host*, applied to a single VirtualSystem, specifies that all the VMs created from that VirtualSystem must be scheduled on different hosts, thus paving the way for high availability configurations.

The new *minimum_LoA* is a QoP term that specifies the minimum Level of Assurance (in authentication) that the resource requires, or the user defines for others to access their resource. Sensitive data will require a higher LoA: the federation authentication code sets the user's LoA when they authenticate; the required minimum is set at negotiation time and is also transmitted as an attribute when the authorisation checks run. Access to resources is blocked if *LoA* $<$ *minimum_LoA*. LoA is discussed further in section 5.1.

Table 1. SLA terms in Contrail

SLA term	Description
vm_cores	number of cores assigned to a VM
memory	RAM size assigned to a VM
cpu_speed	CPU frequency assigned to a VM
cpu_load	average system load of a VM over a 5-minute period
location	country code where the VM is located
availability	% of uptime of the whole provider infrastructure
location	country code where the given shared storage volume is located
storage_reliability	indicates the number of replicas for the given shared storage volume
reserve	number of VMs to be reserved for the given VirtualSystem
co_locate_rack	all the VMs from the given VirtualSystem must be allocated on the same rack
not_co_locate_host	all the VMs from the given VirtualSystem must be allocated on different hosts
minimum_LoA	minimum Level of Assurance that a user wants to be required for accessing her own resources

4.5 Federation Model in Optimis

The OPTIMIS toolkit was developed to support two kinds of providers: the Service Provider (SP) and the Infrastructure Provider (IP). Consequently, the OPTIMIS toolkit supports two different "flavours" of federation:

- a multi-cloud, where the SP uses resources from different cloud providers to deploy a service. This federation is visible to the SP's customer when it selects a service of the SP.
- a cloud federation, which is organised by the IP as part of its internal business processes. Thus, the federation is transparent for the SP and for reasons of data protection SLAs need an explicit term to allow or disallow using IP-federated resources for a concrete service of an SP.

4.6 Role of SLAs in OPTIMIS

SLAs in OPTIMIS are used to describe the SP's requirements on the services offered by an IP [1]. Therefore the SLAs are more technical than those an end-user would see. The SP's requirements are gathered in an XML document, the *Service Manifest*, which is used by the IP to assess whether the requirements can be

[1] The OPTIMIS toolkit is not targeting end-users, thus, and end-user SLAs are not in the scope of OPTIMIS.

fulfilled, have to be negotiated, or cannot be fulfilled at that time. When requirements can be met, the IP converts the Service Manifest into an agreement template for the SP which is then used to create an agreement (possibly after a round of negotiations). The Service Manifest is composed of three parts, the manifest core, which is used for creating the SLA, and the IP and SP extensions. Figure 2 presents the overall structure of the manifest. For a detailed description of the Service Manifest and its API see [22].

If the SP is federating resources, it can negotiate directly with IPs to figure out which are the most suitable Alternatively, the SP can ask a *broker* to organise the federation on its behalf. The broker receives the manifest, selects the suitable IPs, and creates SLAs with all the selected IPs and the SP.

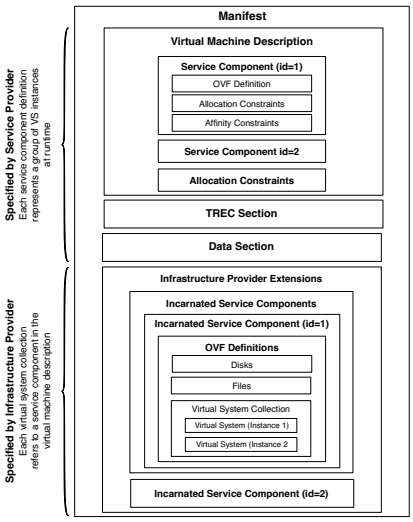


Fig. 2. Example of an infrastructure provider extension in the manifest

4.7 Proposed SLA Terms

The OPTIMIS SLA is based on WSAG4J which in turn is a a reference implementation of WS-Agreement and WS-AgreementNegotiation. Like Contrail, OPTIMIS both reused terms from other projects and developed its own. The latter are describe *trust*, *risk* (of SLA violation), *eco-efficiency* and *cost* (called TREC parameters). Trust is used like Contrail's LoA: An IP starting a federation selects other IPs with a trust level meeting the required trust level specified by the SP. A detailed discussion of the TREC parameters can be found in [23].

The pseudo-XML in Figure 3 shows examples of terms used in Service Manifests, the first showing elasticity, and the second showing eco-efficiency (as usual, '?' denotes that the preceding term is optional, '+' denotes one or more repetitions, and '*' zero or more.)

OPTIMIS also defined QoP terms, specifically data protection and data security, such as geographical locations of the data centres and required level and method for data encryption. Like Contrail, terms for co-location (or non-co-location) are used, which can require same/different rack or same/different datacentre.

5 Comparison and Discussion

The use of automated agents on the service provider side is a classic feature of the cloud (e.g., [18]): the provider does not need to negotiate with each customer

individually, so can provide services to a large number of customers, all of whom are treated the same. In turn, the provider saves money by providing the same services to many customers, by the economy of scale, at the cost of having to be elastic: the number of customers is not known beforehand.

The introduction of an automated agent acting on behalf of the users changes the cloud model. Ultimately, with an agent which is capable of discovering, negotiating, selecting, making use of, and combining services makes the consumer/provider relationship more dynamic. There are underlying assumptions: that the providers are interoperable and interchangeable, that they publish their SLA terms consistently and accurately, that they have consistent frameworks for authentication, authorisation, accounting, and monitoring. It may require relocatable services, or even a full federation built on top of the SLA negotiations, to enable users to access multiple providers with a single credential. But the potential gains are important.

```
<opt:ElasticitySection>
  <opt:Rule>
    <opt:Scope>
      opt:ScopeArrayType
    </opt:Scope>
    <opt:KPIName>xs:string</opt:KPIName>
    <opt:Window>xs:duration</opt:Window>
    <opt:Frequency>
      xs:positiveInteger
    </opt:Frequency>
    <opt:Quota>xs:positiveInteger</opt:Quota>
    <opt:Tolerance>
      opt:PositiveDecimal
    </opt:Tolerance>  ?
  </opt:Rule>  +
</opt:ElasticitySection>
<opt:EcoEfficiencySection>
  <opt:Scope>opt:ScopeType</opt:Scope>
  <opt:LEEDCertification>
    opt:LEEDCertificationConstraint
  </opt:LEEDCertification> ?
  <opt:BREEAMCertification>
    opt:BREEAMCertificationConstraint
  </opt:BREEAMCertification> ?
  <opt:EuCoCCompliant>
    xsd:boolean
  </opt:EuCoCCompliant> ?
  <opt:EnergyStarRating>
    opt:EnergyStarRating
  </opt:EnergyStarRating> ?
</opt:EcoEfficiencySection>
```

Fig. 3. Structure of terms used in Service Manifests

Smaller clouds – unlike the traditional cloud scenario, the customer/provider relationship can become more like the grid, where a single customer can use a significant proportion of the provider's capacity (of course, they don't *have* to, it's just that it is possible). The SLA fixes the customer/provider (or IP/SP) relationship and partially removes the need for elasticity, so the usage/capacity ratio does not need to be low. If a provider has an internal capacity of 100PB (say), and the consumer asks for 50PB, it doesn't matter that the consumer cannot double their capacity. In a federation, they can get the remaining capacity from a different provider. The provider knows that the consumer has allocated 50PB and can plan accordingly (for the duration of the agreement).

Potentially cheaper services – the provider can be less elastic than in a traditional cloud, and the planning ahead also means that the pricing of the service can be set lower, at least for long term agreements. Conversely, the customer's agent can look for lower prices from the competitors, and can even monitor the offers for spot pricing.

Better service delivery – as the agent can choose services from multiple providers, it can potentially combine these to obtain better QoS, while carefully tracking the QoP.

The potential gains are interesting, but should also be "reality checked." True mobility can be achieved only if services can be moved from one provider to another. A high level of interoperation is required. The requirements used by the user agents needs to be sufficiently expressive, but must not make it so complex that they won't be able to describe their requirements. Furthermore, the interplay between requirements needs to be better understood. And, above all, the publication of service terms needs to be timely, consistent, and accurate.

5.1 Quality of Protection

We have discussed a Quality of Protection, to be negotiated and agreed alongside the QoS. Unlike bandwidth, terabytes, or CPU hours, the terms in QoP are less well defined, although there are standards which can be built on. The terms of interest to our projects include:

LoA – the Level of Assurance associated with the user's credential. Covering this in depth is beyond the scope of this paper, but it needs to include factors like cryptographic strength, level of trust in identity provider, unique naming, multifactor authentication, usability, *etc.* Although multidimensional, the LoA is usually summarised in a single value. Typically, the *owner* of the resource or service will set the required minimum LoA based on an analysis of risks. Note that the negotiations of the SLAs themselves need a certain LoA, if we are to assert the trustworthiness of the agreement and the non-repudiation of it. With mutual authentication of sufficient LoA, the protection is also mutual: the customer is protected against a rogue provider, and the provider is protected against misuse by the customer because the customer has signed the agreement.

AUP – the Acceptable Use Policy could usefully be tied in with the SLA: by entering into the agreement, the customer asserts that they will comply with the rules and policies set out by the service provider(s). However, the challenge here is to define when it would be acceptable for an automated agent to enter into such an agreement: as before, the user-defined requirements will need to be clear enough that the agent can determine whether the proposed work is in violation of an offered AUP or not, and the offered AUP needs to be machine readable.

Data protection – again, a full discussion of data protection is beyond the scope of this paper, and, perhaps, beyond the capabilities of most non-specialised clouds. For now, the focus is primarily on availability (file is available at any given time) and integrity (it is not lost and not corrupted). Confidentiality is usually covered by encrypting in flight. Protecting personal data adds another level of difficulty: the best we can do in Contrail and OPTIMIS is to use geographic constraints, or have the providers advertise "no personal data."

6 Conclusion

In this paper we described two approaches for SLA-controlled Cloud federation in the two European projects FP7 projects Contrail and OPTIMIS. While the projects evolved independently using different frameworks for SLA creation and negotiation it turned out that the requirements regarding the SLA processing are similar and the underlying standards used are the same. Further work in the context of SLA negotiation will focus on achieving and testing interoperability of the two approaches.

Acknowledgements. This research was carried out as part of the Contrail and OPTIMIS research projects, funded by the European Commission under the 7th Framework Programme, grant agreements no. 257438 (Contrail) and 257115 (OPTIMIS).

References

1. Ruiz-Alvarez, A., Humphrey, M.: An Automated Approach to Cloud Storage Service Selection. In: ScienceCloud 2011, San Jose, California, USA, June 8 (2011)
2. Ferrer, A.J., et al.: OPTIMIS: A holistic approach to cloud service provisioning. Future Generation Computer Systems 28(1), 66–77 (2012)
3. SLA@SOI project homepage. Website, `http://sla-at-soi.eu/` (visited June 2013)
4. Contrail project homepage. Website, `http://contrail-project.eu/` (visited June 2013)
5. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S.: Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability. In: Proc. of ICIW 2009, 4th Int'l Conf. Web App. and Services, pp. 328–336 (2009), doi:10.1109/ICIW.2009.55
6. Verissimo, P., Bessani, A., Pasin, M.: The TClouds Architecture: Open and Resilient Cloud-of-Clouds Computing. In: Proc. 2nd Int'l Workshop on Dependability of Clouds (DCDV 2012), together with IEEE/IFIP DSN 2012 (2012)
7. Contrail project, Deliverable D2.2, Architecture Design and QoS constraints matching algorithms in Federations (September 2011), `http://contrail-project.eu/documents/18553/136157/D2.2.pdf` (visited June 2013)
8. Buyya, R., Garg, S.K., Calheiros, R.N.: SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions, ArXiv 1201.4522 (2012)
9. Pantazoglou, M., Tsalgatidou, A.: A Generic Query Model for the Unified Discovery of Heterogeneous Services. IEEE Transactions on Services Computing 6(2) (2013)
10. Open Grid Forum. Website, `http://www.ogf.org` (visited May 2013)
11. Wäldrich, O., Battre, D., Brazier, F., Clark, K., Oey, M., Papaspyrou, A., Wieder, P., Ziegler, W.: WS-Agreement Negotiation Version 1.0. Technical Report GFD.193 (2011), `http://www.ogf.org/documents/GFD.193.pdf` (visited June 2013)

12. Svirskas, A., Wilson, M., Matthews, B., Arenas, A., Mac Randal, D., Gallop, J., Bicarregui, J., Lambert, S.: Towards an Efficient, Reliable and Collaborative Web: from Distributed Computing to Semantic Description, Composition and Matchmaking of Services, http://epubs.cclrc.ac.uk/bitstream/755/W3C-FSWS-CCLRCPositionPaper-Svirskas_mdw-v3[1].pdf (visited June 2013)
13. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Technical Report GFD.192 (October 2011), http://www.ogf.org/documents/GFD.192.pdf (visited June 2013)
14. Distefano, S., Puliafito, A.: Cloud@Home: Toward a Volunteer Cloud
15. Cuomo, A., Di Modica, G., Distefano, S., Puliafito, A., Rak, M., Tomarchio, O., Venticinque, S., Villano, U.: An SLA-based Broker for Cloud Infrastructures. J. Grid Comp. 11(1) (2013)
16. Graham-Rowe, D.: BOINC Puts Idle PCs to Work. New Scientist 180(2426-8) (2003) ISSN 0262-4079
17. Hewitt, E.: Java SOA Cookbook. O'Reilly (2009)
18. NIST Standards Acceleration to Jumpstart Adoption of Cloud Computing, http://www.nist.gov/itl/cloud/sajacc.cfm (visited June 2013)
19. Andreozzi, S., Burke, S., Field, L., Galang, G., Konya, B., Litmaath, M., Millar, P., Navarro, J.P.: GLUE Specification 2.0, Open Grid Forum GFD.147, http://www.ogf.org/documents/GFD.147.pdf (visited June 2013)
20. Badger, L., Bohn, R., Chandramouli, R., Grance, T., Karygiannis, T., Patt-Corner, R., Voas, J.: Cloud Computing Use Cases. NIST ITL Cloud Computing, http://www.nist.gov/itl/cloud/use-cases.cfm (visited June 2013)
21. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: Proc. 10th IEEE Int'l Symp. on High-Performance Distributed Computing, HPDC 2010 (August 2001)
22. OPTIMIS Service Manifest and API, http://www.optimis-project.eu/content/service-manifest-scientific-report (visited May 31, 2013)
23. Lawrence, A., Djemame, K., Wäldrich, O., Ziegler, W., Zsigri, C.: Using service level agreements for optimising cloud infrastructure services. In: Cezon, M., Wolfsthal, Y. (eds.) ServiceWave 2010 Workshops. LNCS, vol. 6569, pp. 38–49. Springer, Heidelberg (2011)
24. Kyriazis, D.: Cloud Computing Service Level Agreements. Technical report, European Commission DG CONNECT, Unit E2 - Software and Services, Cloud (July 2013), http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=2496 (visited July 15, 2013)
25. cybula - High performance pattern recognition systems. Website, http://www.cybula.com (visited June 30, 2013)
26. CompatibleOne-The Open Source Cloud Broker. Website http://www.compatibleone.org (visited June 30, 2013)

Paper VI

OPTIMIS: A holistic approach to cloud service provisioning[6]

Ana Juan Ferrer[a], Francisco Hernández[b], Johan Tordsson[b], Erik Elmroth[b], Ahmed Ali-Eldin[b], Csilla Zsigri[c], Raül Sirvent[d], Jordi Guitart[d], Rosa M. Badia[d], Karim Djemame[e], Wolfgang Ziegler[f], Theo Dimitrakos[g], Srijith K. Nair[g], George Kousiouris[h], Kleopatra Konstanteli[h], Theodora Varvarigou[h], Benoit Hudzia[i], Alexander Kipp[j], Stefan Wesner[j], Marcelo Corrales[k], Nikolaus Forgó[k], Tabassum Sharif[l], Craig Sheridan[l]

[a]Atos Origin, Barcelona, Spain
[b]Department of Computing Science, UmeåUniversity, Sweden
[c]The 451 Group, London, UK
[d]Barcelona Supercomputing Center, Barcelona, Spain
[e]School of Computing,University of Leeds, UK
[f]Fraunhofer – SCAI, Sankt Augustin, Germany
[g]British Telecom, London, UK
[h]National Technical University of Athens, Greece
[i]SAP Research Belfast, UK
[j]High Performance Computing Center Stuttgart, Germany
[k]Institut für Rechtsinformatik, Leibniz Universität Hannover, Germany
[l]Flexiant Limited, Livingston, UK

Abstract: We present fundamental challenges for scalable and dependable service platforms and architectures that enable flexible and dynamic provisioning of cloud services. Our findings are incorporated in a toolkit targeting the cloud service and infrastructure providers. The innovations behind the toolkit are aimed at optimizing the whole service life cycle, including service construction, deployment, and operation, on a basis of aspects such as trust, risk, eco-efficiency and cost. Notably, adaptive self-preservation is crucial to meet predicted and unforeseen changes in resource requirements. By addressing the whole

[6]By permission of Elsevier B.V

service life cycle, taking into account several cloud architectures, and by taking a holistic approach to sustainable service provisioning, the toolkit aims to provide a foundation for a reliable, sustainable, and trustful cloud computing industry.

Future Generation Computer Systems 28 (2012) 66–77

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

OPTIMIS: A holistic approach to cloud service provisioning

Ana Juan Ferrer [a], Francisco Hernández [b], Johan Tordsson [b,*], Erik Elmroth [b], Ahmed Ali-Eldin [b], Csilla Zsigri [c], Raül Sirvent [d], Jordi Guitart [d], Rosa M. Badia [d], Karim Djemame [e], Wolfgang Ziegler [f], Theo Dimitrakos [g], Srijith K. Nair [g], George Kousiouris [h], Kleopatra Konstanteli [h], Theodora Varvarigou [h], Benoit Hudzia [i], Alexander Kipp [j], Stefan Wesner [j], Marcelo Corrales [k], Nikolaus Forgó [k], Tabassum Sharif [l], Craig Sheridan [l]

[a] *Atos Origin, Barcelona, Spain*
[b] *Department of Computing Science, Umeå University, Sweden*
[c] *The 451 Group, London, UK*
[d] *Barcelona Supercomputing Center, Barcelona, Spain*
[e] *School of Computing,University of Leeds, UK*
[f] *Fraunhofer – SCAI, Sankt Augustin, Germany*
[g] *British Telecom, London, UK*
[h] *National Technical University of Athens, Greece*
[i] *SAP Research Belfast, UK*
[j] *High Performance Computing Center Stuttgart, Germany*
[k] *Institut für Rechtsinformatik, Leibniz Universität Hannover, Germany*
[l] *Flexiant Limited, Livingston, UK*

ARTICLE INFO

Article history:
Received 20 January 2011
Received in revised form
9 May 2011
Accepted 28 May 2011
Available online 21 July 2011

ABSTRACT

We present fundamental challenges for scalable and dependable service platforms and architectures that enable flexible and dynamic provisioning of cloud services. Our findings are incorporated in a toolkit targeting the cloud service and infrastructure providers. The innovations behind the toolkit are aimed at optimizing the whole service life cycle, including service construction, deployment, and operation, on a basis of aspects such as trust, risk, eco-efficiency and cost. Notably, adaptive self-preservation is crucial to meet predicted and unforeseen changes in resource requirements. By addressing the whole service life cycle, taking into account several cloud architectures, and by taking a holistic approach to sustainable service provisioning, the toolkit aims to provide a foundation for a reliable, sustainable, and trustful cloud computing industry.

1. Introduction

Contemporary cloud computing solutions, both research projects and commercial products, have mainly worked on functionalities closer to the infrastructure, such as improved performance for virtualization of computing, storage, and network resources, as well as fundamental issues such as virtual machine (VM) migrations and server consolidation. When higher-level concerns are considered, existing solutions tend to focus only on functional aspects, whereas quality factors, although very important, are typically not considered. In order to move from a basic cloud service infrastructure to a broader cloud service ecosystem, there is a great need for tools that support higher-level concerns and non-functional aspects in a comprehensive manner.

In this work we introduce five higher-level challenges that in our view must be addressed for a wider adoption of cloud computing:

1. Service life cycle optimization.
2. Dependable sociability = Trust + Risk + Eco + Cost.
3. Adaptive self-preservation.

* Corresponding author.
E-mail addresses: ana.juanf@atosresearch.eu (A.J. Ferrer), hernandf@cs.umu.se (F. Hernández), tordsson@cs.umu.se (J. Tordsson), elmroth@cs.umu.se (E. Elmroth), ahmeda@cs.umu.se (A. Ali-Eldin), csilla.zsigri@the451group.com (C. Zsigri), Raul.Sirvent@bsc.es (R. Sirvent), jordi.guitart@bsc.es (J. Guitart), rosa.m.badia@bsc.es (R.M. Badia), karim@comp.leeds.ac.uk (K. Djemame), wolfgang.ziegler@scai.fraunhofer.de (W. Ziegler), theo.dimitrakos@bt.com (T. Dimitrakos), srijith.nair@bt.com (S.K. Nair), gkousiou@telecom.ntua.gr (G. Kousiouris), kkonst@telecom.ntua.gr (K. Konstanteli), dora@telecom.ntua.gr (T. Varvarigou), benoit.hudzia@sap.com (B. Hudzia), kipp@hlrs.de (A. Kipp), wesner@hlrs.de (S. Wesner), corrales@iri.uni-hannover.de (M. Corrales), nikolaus.forgo@iri.uni-hannover.de (N. Forgó), tsharif@flexiant.com (T. Sharif), csheridan@flexiant.com (C. Sheridan).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.05.022

4. Multi-cloud architectures.
5. Market and legislative issues.

Notably, these five concerns cover most of the ten key obstacles to growth of cloud computing identified in a recent report [1], as well as address several open issues [2–4]. When approaching these concerns, we focus on a holistic approach to cloud service provisioning and argue that a single abstraction for multiple coexisting cloud architectures is imperative for a broader cloud service ecosystem. Thus, our work is based on the assumptions that clouds will be available as private and public, that they will be used in isolation or in a variety of conceptually different combinations, and that they will be internal or external to individual organizations or cross-organizational consortia. The outcome of our multidisciplinary research within these five challenges are incorporated in the *OPTIMIS Toolkit*. We present the design of the toolkit and discuss how it addresses the higher-level concerns introduced here.

The main stakeholders throughout this work are service providers and infrastructure providers, although it can be foreseen that our results can also impact actors such as brokers, and service consumers (end-users). Henceforth, we consider the following definitions for these roles:

- *Service Providers (SPs)* offer economically efficient services using hardware resources provisioned by infrastructure providers. The services are directly accessed by end-users or orchestrated by other SPs.
- *Infrastructure Providers (IPs)* offer computing, storage, and network resources required for hosting services. Their goal is to maximize their profit from tenants by making efficient use of their infrastructures, possibly by outsourcing partial workloads to partnering providers.

The element of interaction between SPs and IPs is a service. Notably, SPs and IPs have conflicting economical and performance goals that result in interesting problems in cases where they are both part of the same organization. It is foreseen that both types of providers are in need of more feature-rich analysis and management tools in order to provide economically and ecologically sustainable services throughout the whole service life cycle.

The outline of the paper is as follows. Sections 2–6 present details of the five higher-level concerns enumerated above. Section 7 presents a high-level view of the OPTIMIS Toolkit, discusses how the toolkit addresses the five challenges, and illustrates how it can be used to instantiate various cloud architectures. Experimental results are described in Section 8. Finally, we share our concluding remarks in Section 9.

2. Service life cycle optimization

There are three fundamental steps in the service life cycle, construction of the service, deployment of the service to an IP, and operation of the service.

2.1. Service construction

In the service construction phase, the SP builds the service and sets it up for deployment and operation on the IP. The activities performed include preparation and configuration of VM's images as well as specification of dependences among service components. Currently, there is no programming model specifically tailored for clouds. On the one hand developers are limited to use application-specific platforms [5], restrictive computing paradigms [6,7], or platforms for a single cloud middleware [8]. A common way of offering these solutions is by wrapping them as a PaaS environment or by offering proprietary APIs for particular middlewares limited to single infrastructure providers. On the other hand, developing high-level services from raw infrastructure through use of IaaS is a manual and ad-hoc process, hindering broader cloud adoption as service development becomes expensive and time consuming.

The challenge of service construction resides in designing and developing easy ways to create complex services [9]. To this end, applications need to be abstracted from their execution environment and the development of new services, including those composed from adapting and combining legacy-and licensed software, must be facilitated. For the latter, novel license management technologies are required to significantly extend currently available solutions for management of license tokens in distributed environments [10]. The composition of services as a mix of software developed in-house, existing third-party services, and license-protected software is a clear contrast to commonly used approaches for service composition [11].

2.2. Service deployment

In the service deployment phase, the service is placed on an IP for operation. From the SP point of view, the main objective during this phase is to select the most suitable IP for hosting a service, whereas for IPs the main objective is to decide whether accepting a new service is beneficial for its business goals. The key process during deployment is the negotiation of SLA terms between SP and IP. However, this negotiation is performed manually in current deployment solutions [2] and SPs are limited to use single providers as differences in contextualization mechanisms [12,13], necessary for instantiating a service once deployed, hinder multi-cloud deployment.

Moreover, contemporary cloud SLA mechanisms [14,15] are typically limited to cost-performance tradeoffs. For example, it is not possible to automatically evaluate levels of trust and risk, or to negotiate use of license-protected software. To overcome current limitations, deployment optimization tools need to support deployment given a set of policies and allow SPs to specify required SLA terms for services. The policies governing deployment must include the degree of trust expected from a provider, the level of risk with regard to cost thresholds, energy consumption limits, performance levels, etc.

2.3. Service execution

Service execution is the last phase in the service life cycle and consists of two different but related procedures, performed by SPs and IPs. The overall objectives of these stakeholders differ and as a result there is a conflict of interest in management tasks. On one hand, the SP performs a set of management operations in order to meet the high-level Business Level Objectives (BLOs) specified during service construction. These include, for instance, constant monitoring of service status and mechanisms for monitoring and continuous assessment of the risk level of IPs in order to apply the corresponding corrective actions. On the other hand, an IP performs autonomic actions to, e.g., consolidate and redistribute service workloads, replicate and redistribute data sets, and trigger actions to increase and decrease capacity to adhere to SLAs, i.e., enact elasticity rules, with the overall goal of achieving the most efficient use of its infrastructure and hence maximize its own objectives, potentially at the expense of the goals of the SPs.

Contemporary tools for service execution optimization focus on mechanisms for monitoring service status and for triggering capacity variations to meet elasticity requirements [16]. These tools tend to use only SLAs and infrastructure status for making decisions and either neglect business-level parameters such as risk, trust, reliability, and eco-efficiency, or consider them in isolation. For instance, eco-efficient policies for the operation of hosting centers aiming to minimize its power consumption have

been investigated [17,18]. Similarly, trust mechanisms have been studied in the context of grid resource selection in order to choose providers that are likely to provide better service according to their reputation [19]. In the same way, risk information has been used for decision making [20]. Finally, some resource management proposals for data centers and e-commerce systems are driven by business objectives or incorporate business level parameters in their management policies [21–23], though most of them target revenue as the only objective.

According to this, SPs and IPs require software components that in addition to the traditional performance indicators also take into account business-level parameters (e.g., risk, trust, reliability, etc.) in order to make decisions in a synergistic fashion to contribute to the overall provider goals. In order to achieve this type of decision making process, all management activities must be harmonized through the use of cloud governance processes that integrate all service requirements, from high-level BLOs to infrastructure requirements.

3. Dependable sociability = trust + risk + eco + cost

Traditionally, relationships between stakeholders have been focused on cost-performance trade-offs. However, economical factors are not enough for an open and highly dynamic environment in which relationships are created in an on–off basis with a possible high degree of anonymity between stakeholders. On the one hand, it is necessary to offer methods and tools to quantitatively assess and evaluate stakeholders, e.g., through audit and monitoring functions to analyze the probability of service failure, the risk of data loss, and other types of SLA violations. On the other hand methods to measure stakeholder satisfaction such as individual and group perceptions, reputation [24] of stakeholders regarding ecological aspects, or previous experiences, must also be considered. Altogether, these mechanisms can be used to confirm the dependability and reliability among stakeholders. We now introduce various quality factors that have traditionally not been considered but that we believe improve the decision making capabilities of both SPs and IPs.

3.1. Trust—reputation management

Trust is a multifaceted aspect not only related to risk and security aspects, but also to perceptions and previous experiences. Selection of an IP depends on the trust that it will provision the service correctly and securely. Conversely, knowing a customer's reputation enhances the admission control evaluation and reduces the risk of breaking economical or ecological goals of an IP.

Trust is often calculated by reputation mechanisms [25,26]. A reputation is a subjective measure of the perception that members of a social network has of one another. This perception is based on past experiences. The reputation ranks aggregate experiences of all members of the social network—in this case the social network includes all stakeholders, i.e., the combination of SPs and IPs. Tools to determine the integrity of data disclosed by stakeholders as well as mechanisms to act accordingly, e.g., to blacklist dishonest providers, are also necessary.

3.2. Risk assessment

Underpinning a successful cloud infrastructure is delivering the required QoS levels to its users in a way that minimizes risk, which is measured in terms of a combination of the likelihood of an event and its impact on the provision of a functionality. Earlier work in risk management for distributed systems has mainly focused on operational aspects such as failures and performance degradation, and assumed a very IaaS centric view under a specific resource reservation model [27]. Factors such as trust, security, energy consumption, and cost have not been traditionally considered. Moreover, tools for the definition, assessment and management of risk based on variations in levels of the proposed factors for both stakeholders and throughout the service life cycle (SPs during service construction, deployment, and operation; and IPs during admission control and internal operations) are virtually nonexistent. Such risk management mechanisms must also consider aspects such as energy consumption, the cost of reconfiguration and migration, and the reliability and dependability of the provided services, in order to maintain secure, cost-effective, and energy-efficient operations.

3.3. Green assessment

Environmental concerns reflected in upcoming legislation have increased the awareness of the ecological impact of the ICT industry. As a result, the level of ecological awareness can now be a deciding factor between competing providers. However, environmental concerns are not the only reason for the growing interest in green data centers, rising electricity prices can also guide the deployment of services to locations in which they are provisioned in a more efficient way. The consequence is that IPs must now focus more than ever on energy efficiency aspects.

Cloud computing in itself contributes to reduce power consumption by consolidating workloads from different customers in a smaller number of physical nodes, turning off unused nodes [28]. The key concern to confront is the tradeoff between performance and power consumption [29–31], i.e., to minimize power consumption but still accomplish the desired QoS [32]. To address this tradeoff, energy efficiency must be treated equal as the other critical parameters, already including service availability, reliability, and performance. To this end, a broad set of mechanisms are required, ranging from tools for logging and assessing the ecological impact at the service level, theoretical models that characterize the power consumption of services depending on configuration parameters (e.g., clock frequency, resource usage, and number of threads used), to predicting future energy impact based on run-time state, historical usage patterns, and estimates of future demands.

3.4. Cost and economical sustainability

Fulfilling high trust levels between stakeholders, reduced risk, and eco-efficient provisioning is trivial if cost is not an issue. Economical aspects are necessary to balance the previous three goals. Furthermore, cost must be an explicit parameter throughout the entire service life cycle. Current commercial providers offer a variety of capabilities under different pricing schemes, but it is hard to differentiate among the offerings without sufficient knowledge of the repercussions on internal performance, ecologic, and economic goals. To improve this situation, more complex economic models are needed. These models must include features to compare economical repercussions between alternative configurations. To this end, such models must employ business related terms that can be translated to service and infrastructure parameters during development and deployment of services. During the operation of a service, it is necessary to optimize economical factors through a combination of runtime monitoring, analysis of historical usage patterns, and predictions of future events. The latter helps to anticipate future service economic trends. All of these actions create an economic policy framework in which stakeholders can specify the autonomic behavior expected from the elements under their management responsibility.

4. Adaptive self-preservation

Service and infrastructure management in clouds is difficult due to the ever-growing complexity and inherent variability in

services. Rapid responses to events are necessary to satisfy agreed SLAs. Accordingly, human administration becomes unfeasible and building self-managed systems seems to be the only way to succeed.

Although self-management for cloud infrastructures is a novel research area, approaches for automated adjustment of resource allocations for virtualized hosting centers can be applied [33,34]. However, many of these approaches have two main limitations. First, they typically exhibit a lack of expressiveness in self-management due to a lack of a holistic view of management. This results in management actions, e.g., resource allocation, monitoring, or data placement, that are performed in isolation and are as such not optimal. Second, existing solutions tend to use only SLAs and infrastructure status for making decisions, neglecting business-level parameters such as risk, trust, reliability, and eco-efficiency, or considering them in isolation, as discussed in Section 2.3.

To overcome this problem, SPs and IPs require that all management actions are harmonized by overarching policies that incorporate, balance, and synergize aspects of risk assessment, trust management, eco-efficiency, as well as economic plausibility. The management actions must be handled by software components able to monitor and assess their own status and adapt their behavior to ever changing conditions. Aspects to consider in this decision are overall BLOs, infrastructure capabilities, historical usage patterns, and predictions of future demands. The result must be an integrated solution capable of a wide range of autonomic management tasks [35] including self-configuration, i.e., automatic configuration of components, self-healing, i.e., automatic discovery and correction of faults, and self-optimization, i.e., automatic optimization of resource allotments and data placement. Example autonomic management tasks include SLA enforcement, recovery of service operation upon resource failure, VM placement optimization (including migration [36]), enactment of elasticity policies (vertical and horizontal scalability), consolidation of services, management of advance reservations, and data replication for fault tolerance or performance improvements.

5. Multi-cloud architectures

There are at least two fundamentally different architectural models for cloud service provisioning using multiple external clouds:

- In a *federated cloud*, an IP can sub-contract capacity from other providers as well as offer spare capacity to a federation of IPs. Parts of a service can be placed on remote providers for improved elasticity and fault tolerance, but the initial IP is solely responsible for guaranteeing the agreed upon SLA (with respect to performance, cost, eco-efficiency, etc.) to the SP.
- In a *multi-provider hosting* scenario, the SP is responsible for the multi-cloud provisioning of the services. Thus, the SP contacts the possible IPs, negotiates terms of use, deploys services, monitors their operation, and potentially migrates services (or parts thereof) from misbehaving IPs. IPs are managed independently and placement on different providers is treated as multiple instances of deployment.

Each model has benefits and drawbacks. However, to date, these models have only been studied in isolation [13,16,37], which essentially creates either–or situations. Instead, for a more flexible provisioning model, it is important to be able to use multiple clouds without distinguishing whether a service is hosted within a single cloud or across multiple providers, i.e., clouds must be able to be combined into arbitrary, hierarchical architectures. To this end, it is imperative to create a single abstraction without regard of architectural style. To accomplish this, there are a number of challenges that must be solved, including: verification of SLA adherence; metering, accounting and billing of services running out of a provider's boundaries; managing software license authorizations, particularly when migrating a service to different providers; replication, synchronization, and backup of data between providers; evaluation of economical efficiency associated with using external providers; legal implications regarding data protection and privacy aspects; and establishing an inter-cloud security context for governing all interactions between clouds.

6. Market and legislative issues

Clouds bring change to user behavior. The focus of attention is moving away from how a service is implemented or hosted to what the service offers, a shift from buying tools that enable a functionality to contracting third-party services that deliver this functionality on demand in a pay-per-use model [38]. There is a massive surge in interest around private and hybrid clouds. With new application use cases emerging on a regular basis, numerous commercial on-ramps are seeking to provide access to multiple clouds, and startups and incumbent providers alike are targeting cloud service brokerage. These changes in the landscape create opportunities for new roles, relationships, and value activities.

We believe that the hybrid cloud is where the market is heading. The appetite among enterprises for a range of execution environments to serve the needs of different workloads reinforces that successful cloud strategies will enable the *best execution venue* practices supported in hybrid cloud environments. These allow service providers to choose, via policy automation, different venues (private, public, federated, etc. clouds) in which to run workloads, depending on costs, latency, security, locality, eco-efficiency, or other SLA requirements. In short, we work toward a cloud market model where providers are fungible, transparent and compliant, and consumers can easily and efficiently use cloud functionality to their best advantage.

In addition to the new market opportunities, the emerging cloud landscape introduces additional concerns related to legal compliance. It is important to assess from the very beginning those associated legal risks in cloud computing and create a framework for minimizing or mitigating those risks, particularly when presupposing that data moves geographically. In such cases, data protection and privacy, being issues of cross-border jurisdictional nature as they concern the acquisition, location, and transfer of data [39], are important and call for a data protection framework and security infrastructure [40,41]. Furthermore, legally and non-legally binding guidelines concerning green IT strategies and legislative and jurisdictional issues are key to infrastructure and service providers when it comes to decision making [42,43]. In addition, intellectual property and contractual issues concerning ownership and rights in information and services located in the cloud need to be tailored and taken into account when designing a cloud computing toolkit [44].

7. The OPTIMIS toolkit

Our response to the challenges presented in the previous section is a multi-disciplinary research line with inclusion of the main outcomes in the OPTIMIS Toolkit. The toolkit consists of a set of fundamental components realizing an anticipating a variety of architectures for simultaneous use of multiple clouds. Fig. 1 illustrates the high-level components of the toolkit: the *Service Builder (SB)*, the *Basic Toolkit*, the *Admission Controller (AC)*, the *Deployment Engine (DE)*, the *Service Optimizer (SO)*, and the *Cloud Optimizer (CO)*.

The SB is used during the service construction phase and enables developed services to be delivered as Software as a

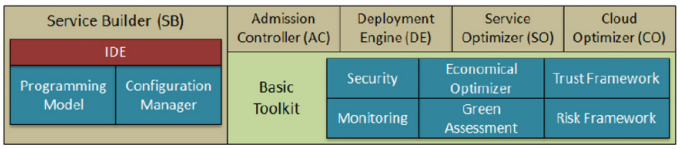


Fig. 1. High level view of the components in the OPTIMIS toolkit. The basic toolkit addresses quantitative and qualitative analyzes that help in making optimal decisions regardless of the invoking component. Organizations can act as SPs and/or IPs depending on the components that they chose to adopt (from the admission controller, deployment engine, service optimizer, and cloud optimizer).

Service (SaaS). A service programmer has access to an integrated development environment that simplifies both development and configuration of the service using a novel programming model for service development. In our programming model, a service is a collection of *core elements*, that include services built from source code, existing services, licensed software, and legacy-software not developed specifically for clouds, as well as a set of dependences between these core elements. During operation of the service, the core elements are orchestrated by a runtime environment that analyzes the dependences defined during service construction.

Each core element has a set of functional and non-functional requirements associated, e.g., requested performance, amount of physical memory and CPU characteristics, response time, elasticity aspects, service and security level policies, ecological profile, etc. In addition, there are also requirements among the core elements and between the service and its potential users. Once the core elements are implemented, these are packed by the SB along with any external software components into VM images. The configuration of these VM images are then encoded in a service manifest based on the Open Virtualization Format (OVF) [45] with a set of extensions to specify the functional and non-functional requirements of the service. This service manifest is the input to the service deployment phase, and the completion of the manifest marks the end of the service construction phase.

The Basic Toolkit provides functionalities common to components that are used during service deployment and execution. Some of the functionalities address the quantitative and qualitative requirements that we in Section 3 discuss under the term Dependable Sociability. Monitoring and security are functionalities that must be considered during several stages of the service life cycle. The Basic Toolkit provide general purpose functionalities that evaluate similar aspects of management. However, component behavior is customized depending on the invoking module. This customization is fulfilled through the use of internal policies that adapt the decision making processes, e.g., based on the invoking component and the current stage of the service life cycle.

For example, to create a comprehensive trustworthy system, the toolkit considers all relationships of the types SP–IP and IP–IP. Trust estimations are determined from the trust rank of the SP (or IP) in other members of its own social network according to a transitive trust chain. The reputation mechanism delivers trust measurements at two levels: for IPs, trust reflects their performance and the ability to accomplish promised levels of service, whereas for SPs, trust measurements are relevant for establishing successful business networks. For IPs, the trust tools include methods to identify SPs in long term relationships and to analyze SPs' historical behavior that can help to improve management operations by e.g., predicting future capacity.

We now illustrate how the OPTIMIS Toolkit is used during deployment of services. The first scenario is a simplified one where an IP delivers capacity to a SP as shown in Fig. 2. More complex scenarios can also be realized as described later in the section. Service deployment starts from where service construction ends, with a service manifest that describes the VM images and associated requirements for service configuration.

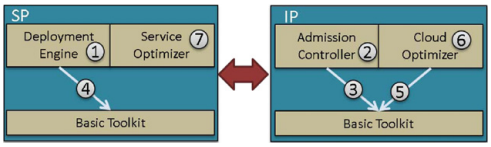


Fig. 2. Service deployment scenario that illustrates the interaction between the high-level components of the SP and IP.

During service deployment the SP finds, by use of the DE, the best possible conditions for operation of the service, negotiates terms of deployment, and launches the service in the IP. The DE send the service manifest (SLA template) to the IP to receive deployment offers (*Step* 1 in Fig. 2).

An IP receiving a deployment request performs a probabilistic admission control to decide whether to admit the new service or not (*Step* 2). This test balances revenue maximization lead by business goals against penalties for misbehavior, i.e., from breaking SLAs of currently provisioned services. Example policies include *over-provisioning* (overbooking for revenue optimization) as well as *under-provisioning* (reserving capacity to minimize the risk of failures). The test is carried out by the Admission Controller (AC) component with help of use of the Basic Toolkit (*Step* 3). An integral part of the admission control test is workload analysis of the current infrastructure and the new service, to be performed in combination with capacity planning.

Using the Basic Toolkit, the DE evaluates the IP's offers to run the service in order to chose the most suitable one (*Step* 4). This analysis is carried out considering both qualitative and quantitative factors discussed in Section 3. After selecting a deployment offer, the DE prepares the service images for deployment. In this step, information required for the VMs to be able to self-contextualize once they boot is embedded in ISO images that are bundled together with the VM images.

In the IP, the process of accepting a new service starts by allocating space for the VMs and determining their initial placement. The latter is a complicated process as placement must consider the (predicted) elasticity of the service and the non-functional constraints specified in the deployment manifest. Some of these constraints can even have legal ramifications regarding e.g., data protection and privacy or environmental guidelines as discussed in Section 6. Allocation of resources for the service is performed by the CO with help of components for management of VMs and data that both make extensive use of the functionalities in the Basic Toolkit (*Step* 5). Once resources are allocated by the CO with help of these managers, the VM images are booted and self-contextualized with help of previously the scripts installed (*Step* 6). The SO in the SP is notified once the deployment process completes (*Step* 7).

The SO and CO also perform repeated management decisions during service operation, the SO on behalf of the SP and the CO for the IP. The SO continuously checks that the IP provisions the service according to the agreed SLAs, otherwise the SO can migrate the service to a different IP. On the other hand, the CO optimizes the IP's infrastructure resources. This includes, for instance, monitoring

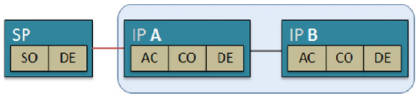


Fig. 3. Federated cloud architecture where the SP establishes a contract with IP A that is a member of the federation that includes IP B. The service is delivered using resources of either IP, or from both.

of infrastructure status, recovery from failures (e.g., by using checkpoints), and mechanisms for optimizing power consumption (e.g., by consolidating and migrating VMs). A complicating factor in this infrastructure optimization is that the CO also dynamically increases and decreases the number of VMs for a service (i.e., performing elasticity management) to protect SLAs with SPs to avoid penalties and preserve reputation. The SO and CO both utilize the Basic Toolkit as the basis for their quantitative and qualitative analysis.

7.1. Flexible multi-cloud architectures

The combination of the five main components of the OPTIMIS Toolkit and their implementation by SPs and IPs, gives rise to a number of plausible multi-cloud scenarios where resources from more than one IP can be combined in novel ways. Some example compositions follow.

Federated architecture

In this scenario (Fig. 3), several IPs (A and B) use the OPTIMIS Toolkit to establish a cooperation in which any IP can lease capacity from the other. The cooperation is carried out according to internal IP business policies. The SP is unaware of this federation as its contract is with a single IP (in this case IP A). However, the SP can indirectly pose constraints on which IPs in the federation that can be used through non-functional requirements such as affinity of service components or juridical restrictions to prevent VM migration across country borders (data protection areas). In the federated scenario, the contracted provider (IP A) is fully responsible toward the SP even in the case of subcontracting of resources from the federation.

Multi-cloud architecture

In this scenario (Fig. 4), the SP is responsible for the multi-cloud aspect of service operation. If IP A does not fulfill the agreed objectives, the SP can cancel the contract and move the service to a different IP (IP B). Notably, the SP is responsible both for negotiating with each IP and for monitoring the IPs during service operation. In more complex variations of this scenario, parts of the service can be hosted on multiple providers. By using APIs and adapters externally to the OPTIMIS components, the toolkit can also achieve interoperability with non-OPTIMIS providers (IP C in Fig. 4). However, in such cases, the SP has to resort to less feature-rich management capabilities, and the risk levels for service provisioning increase accordingly.

Aggregation of resources by a third party broker

This scenario, illustrated in Fig. 5, introduces a new stakeholder, the broker, which aggregates resources from multiple IPs and offers these to SPs. The broker thus acts as a SP to IPs and as an IP to SPs. Given the conflicting goals of these respective providers, there are many interesting concerns regarding the independence, honesty, and integrity of the broker. Benefits with this model for SPs include simplicity and potential cost reductions, as the broker can provide a single entry point to multiple IPs and may obtain better prices due to bulk discounts from IPs. Management is simplified for an IP that offers capacity to a broker as the number of customers is decreased. Accordingly, trust and risk become easier to predict as the IP is likely to have fewer and longer term contracts

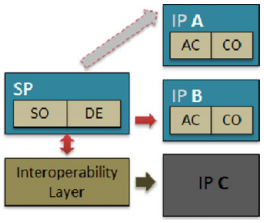


Fig. 4. Multi-cloud architecture in which the SP terminates the contract with IP A and re-deploys the service to IP B. The SP can also use infrastructure from an IP (C) that does not implement the OPTIMIS Toolkit. In this case the SP uses an interoperability layer that is external to the OPTIMIS components.

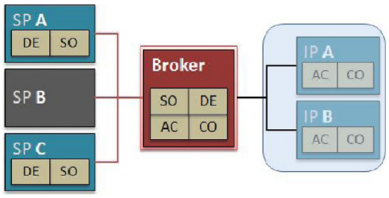


Fig. 5. In the brokering architecture, a third party broker aggregates resources from several IPs (A and B) and offer these resources to SPs.

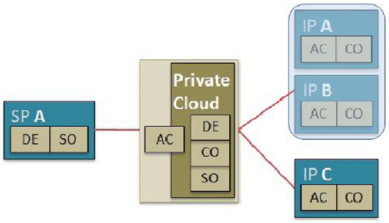


Fig. 6. Hybrid cloud architecture. An organization moves part of its operation to external providers (the federation formed by IPs A and B, as well as C). The organization can also sell capacity to the SP during periods of low load.

with brokers, instead of a multitude of short interactions with potentially unknown SPs.

Hybrid cloud architecture

In this scenario (Fig. 6), any organization that operates a private cloud is able to externalize workloads to public IPs. This is accomplished by monitoring the normal operation of the private cloud, through the CO component, and using capacity from public clouds (IPs A, B, and C, the former two in federation) when the local infrastructure is insufficient. The implementation of this scenario is significantly simplified once an organization is using the OPTIMIS Toolkit for managing its private cloud. Furthermore, this scenario can be extended if the organization makes use of the AC. In this case, the organization is able to offer capacity from its private cloud to others (SP A) when that capacity is not needed for internal operations.

8. Evaluation

This section describes experimental evaluations of selected parts of the OPTIMIS Toolkit, namely how cost and risk can be included in elasticity policies and how to evaluate cloud providers through risk assessment.

Table 1
Comparison of elasticity policies.

Policy	Failed	Cost	Failed (%)	Cost (%)
UR-DR	−17 829	349 648	−0.11	2.2
UR-DP$_{C1}$	−54 296	196 790	−0.34	1.24
UR-DP$_{C2}$	−5 813	566 795	−0.037	3.6

8.1. Elasticity policies for cost-risk tradeoffs

The key aspect for elasticity is to allocate and deallocate the right number of VMs to a service, and at the right time. Over-provisioning, i.e., allocating too many VMs, reduces the risk for (availability) SLA violations, but increases costs as some VMs are not needed. Conversely, allocating too few VMs reduces costs but increases the risk for failing to provision the service in accordance with the SLA. An ideal elasticity policy should be able to proactively and accurately estimate the future service load and thus be able to reduce the SLA violation rate, with a minimal amount of over-provisioning. For elasticity, we define a risk asset to the number of service requests the IP failed to provision proactively. We define the cost to be the number of VMs over-provisioned at any time unit i.e., resources allocated but not used.

An elasticity policy defines two decision points, one for when and how much to scale up and another one for scale down. We here study three different elasticity policies, all based on closed loop control systems. The first and simplest policy scales up and down reactively and is henceforth denoted *UR-DR*. The second policy, denoted *UR-DP*$_{C1}$, scales up reactively, but scales down using a proactive controller, where the gain parameter is based on the periodical change of the system load. Similarly, the *UR-DP*$_{C2}$ policy combines reactive scale up with proactive scale down, where the gain parameter is the ratio between the load change and the average system service rate over time. A more in-depth discussion of control approaches to elasticity is found in our previous work [46].

We evaluate the three elasticity policies using a 17 days and 19 h subset of the web server traces from Wikipedia [47]. In this evaluation we assume that each VM can service 100 requests simultaneously and that load balancing is perfect. The number of VMs suggested by the elasticity policy is thus a fraction of the number of requests anticipated for the Wikipedia site. A summary of the experiments is shown in Table 1. In this table, the Failed column shows the number of VMs needed (but not allocated) to serve all requests, i.e., the degree of under-provisioning. The Cost column shows the number of VMs allocated but not used, i.e., the amount of over-provisioning. In the two right-most columns these two metrics are given as percentages of the total number of VMs.

Fig. 7 shows each service request (marked '+') in Wikipedia workload and the allocated capacity (marked '×'), i.e., number of VMs multiplied by 100. Fig. 7(a), (c) and (e), show the complete 17 days 19 h trace, whereas Fig. 7(b), (d), and (f) detail the number of service requests and allocated VMs for a period of 17 min. As shown in Table 1 and illustrated in Fig. 7, the three policies result in different risk levels associated with different costs. The *UR*−*DC*1 policy provides the lowest cost combined with the highest risk. The *UR-DR* policy results in a medium risk level coupled with a medium cost, whereas *UR-DP*$_{C2}$ provides the lowest risk but at the highest cost. The general trend observable here is that when doubling the over-provisioning from 1.24% to 2.2% the risk is lowered to one third, and when further increasing over-provisioning to 3.6%, risk is cut to around a tenth. We remark that for the Wikipedia traces, both risk and cost are very low, with less than 0.4% under-provisioning and less than 4% resources over-provisioned by any policy. The good results are due to this workload being rather smooth and regular with the exception of a sharp peak at around 600,000 s.

To further understand how risk and cost vary with the workload size, we performed a second set of experiments where the number of service requests varies. In this experiment, we multiply the number of requests per time unit in the Wikipedia traces by a factor from 1 to 10 and by 20, 30, and 40. Figs. 8 and 9 illustrate how the risk and cost, both defined as in the previous experiment, change with the workload size. The *UR-DR* and *UR-DP*$_{C1}$ policies show similar behavior, namely that when the workload size increases, the risk (under-provisioning) increases up to around 0.8% whereas the cost (over-provisioning) decreases to around 1%. On the contrary, the *UR-DP*$_{C2}$ shows a stable behavior, with risk around 0.04% and cost just above 3.5% for all workload sizes. These results suggest that the *UR-DP*$_{C2}$ elasticity policy is more robust and should be used for services with unknown workloads for which low-risk provisioning is desired.

8.2. SP and IP evaluation through risk assessment

The OPTIMIS risk assessor provides the functionality to evaluate providers (*SP*/*IP*) based on a number of criteria. These evaluations are core parts in provider decision making, e.g., for service deployment and admission control. The SP uses the following criteria:

- Past performance: Record with respect to SLA acceptance and violation rate in past SLAs.
- Maintenance: Based on the IP's policy for maintaining their infrastructure.
- Infrastructure: Based on available resources, fault-tolerant mechanisms available, use of redundant network, etc.
- Security: Based on the IP's security policy regarding access to resources.
- Customer support: The IP's policy with regards to customer support, e.g. do they have a 24/7 contact number?

On the other hand, the IP uses the following criteria for assessing an SP:

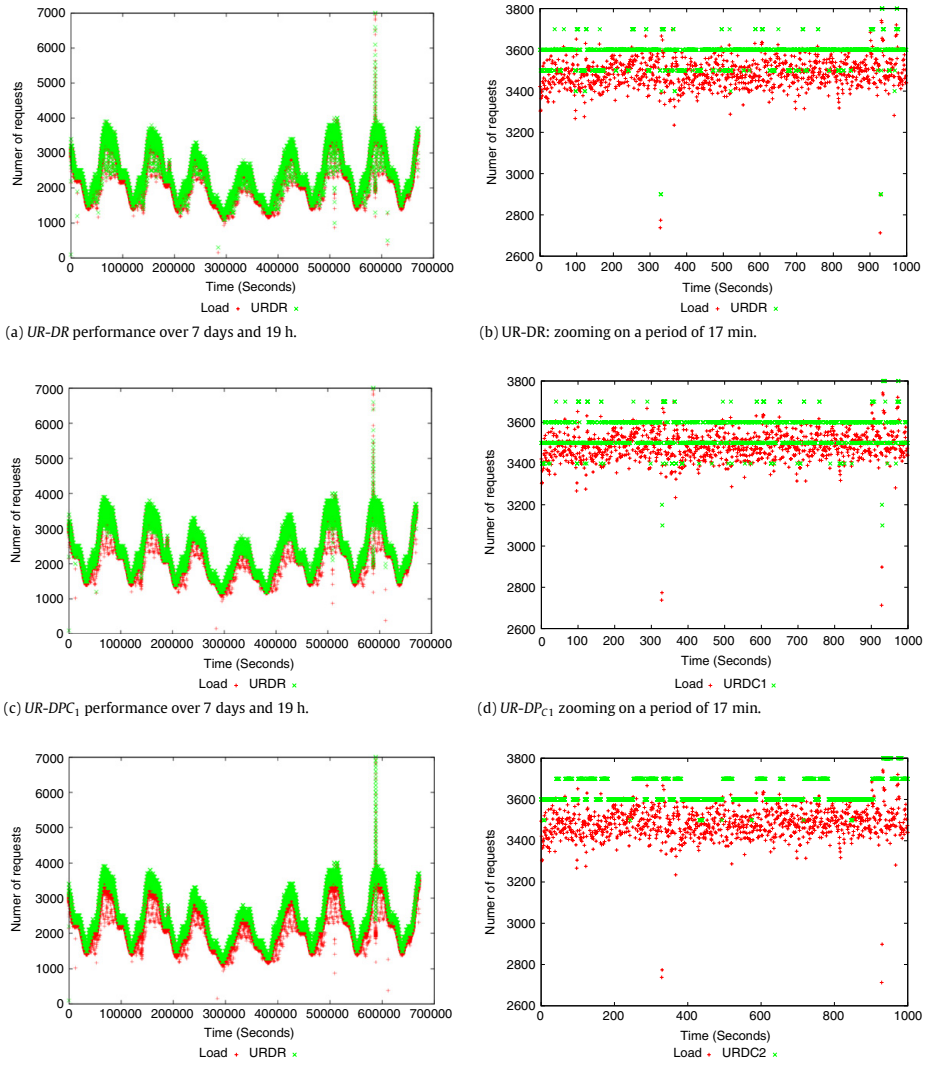
- Past performance: Record with respect to SLA acceptance and violation rate in past SLAs.
- Legal: Based on the SP's policy for supporting legal aspects.
- Security: Based on the SP's security policy regarding the use of resources.

A value between 0 and 1 is computed for each of the criterion by evaluating a provider with respect to a number of sub-criteria. These values are used as the basis for the evaluation. There are two important features of the evaluation system. First, it takes into account provider preferences. Different providers are likely to value the criteria differently. Therefore providers are able to specify the importance of each of the criteria on a scale of 0 to 10. These are then translated into criteria weights that encapsulate the amount of influence a particular criterion should have and incorporated into the provider evaluation. Second, it is able to handle missing data. Some providers may be unwilling to share all of the information necessary to compute the criteria values. Alternatively, data may have been corrupted.

The assessment is achieved through an implementation of Dempster–Shafer Analytical Hierarchy Process (DS-AHP), whereby each decision alternative (in this case each provider) is mapped onto a belief and plausibility interval [48].

Consider a set of providers as corresponding to the proposition that the providers in that set are preferable to all other providers considered in the evaluation but not to each other. The end-user preference weights, w_i, are computed for each criterion, $i = 1 \ldots N$. Pair-wise comparison of decision alternatives (for providers) are used to derive weights for the criteria, $r_j^{(i)}$ for the ith criterion and jth provider. A weight or Basic Probability Assignment (BPA) is computed for each provider as:

$$m_j = \sum_i w_i r_j^{(i)}.$$



(a) *UR-DR* performance over 7 days and 19 h.

(b) UR-DR: zooming on a period of 17 min.

(c) *UR-DPC*$_1$ performance over 7 days and 19 h.

(d) *UR-DP*$_{C1}$ zooming on a period of 17 min.

(e) *UR-DP*$_{C2}$ performance over 7 days and 19 h.

(f) *UR-DP*$_{C2}$ zooming on a period of 17 min.

Fig. 7. Performance of elasticity policies.

If data is missing $r_j^{(i)}$ values cannot be computed. Instead of pair-wise comparison, for any given criterion, each decision alternative (provider) is evaluated relative to the *frame of discernment* (the entire decision space). Providers which are indistinguishable with respect to a criterion are grouped together as a single proposition (that the providers in this group are the best alternative). This results in the BPAs in the form $m_i(s)$ where s is a set of one or more providers. Criteria BPAs are combined using Dempster's rule of combination:

$$(m_1 \oplus m_2)(y) = \frac{\sum_{s_1 \cap s_2 = y} m_1(s_1) m_2(s_2)}{1 - \sum_{s_1 \cap s_2 = \emptyset} m_1(s_1) m_2(s_2)},$$

where $\emptyset$ is the empty set.

Belief in the proposition A is defined as the exact belief that either A or some subset of A is true:

$$Bel(A) = \sum_{\mathbf{B} \subseteq \mathbf{A}} m(B). \quad (1)$$

Here, $m(B)$ is a basic probability assignment for the proposition B [48]. The plausibility of proposition A is a measure of the extent to which we do not disbelieve proposition A and is defined as:

$$Pls(A) = \sum_{\mathbf{A} \cap \mathbf{B} \neq \emptyset} m(B). \quad (2)$$

These form an interval, [Bel(A), Pls(A)] with respect to proposition A. These intervals, for each proposition corresponding to a single provider, are used to compute the order of preference for

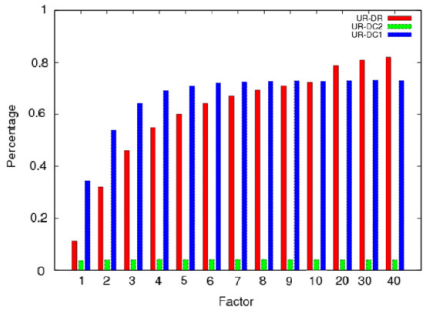


Fig. 8. Effect on risk of varying workload sizes and elasticity policies.

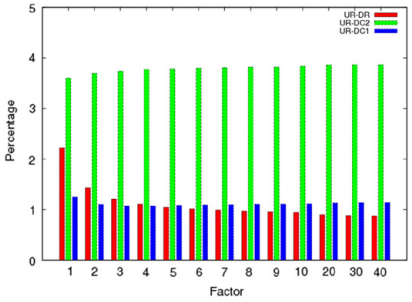


Fig. 9. Effect on cost of varying workload sizes and elasticity policies.

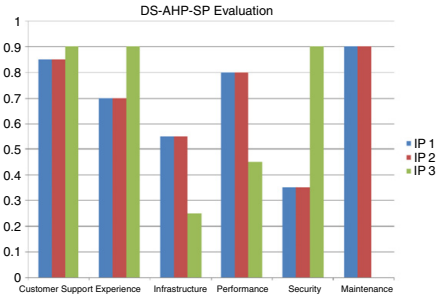


Fig. 10. SP-DS-AHP assessment of three IPs.

the providers. A preference value for provider A relative to provider B is computed as:

$$P(A > B) = \frac{max[0, Pls(A) - Bel(B)] - max[0, Bel(A) - Pls(B)]}{[Pls(A) - Bel(A)] + [Pls(B) - Bel(B)]}. \quad (3)$$

If $P(A > B) > 0.5$ then provider A is preferred. Comparison with simulated data is used in order to obtain a provider ranking between 0 and 1, where 1 corresponds to better than all providers, 0 corresponds to worse than all providers, and 0.5 is average.

Preference values are derived to directly compare the providers. Fig. 10 shows SP's assessment with rating results 0.66, 0.33, and 1 for providers IP_1, IP_2, and IP_3 respectively. Provider 3 is regarded as preferable to both IP_1 and IP_2 with certainty since belief in IP_3 as the best choice is greater than the plausibility of either IP_1 or IP_2. Hence the computed ranking is IP_3, IP_1, and IP_2. Fig. 11 shows IP's assessment with rating results 0.33, 1, and 0.66 for providers SP_A, SP_B, and SP_C respectively, which leads to the final ranking: SP_B, SP_C, SP_A. More details on DS-AHP are found in [48].

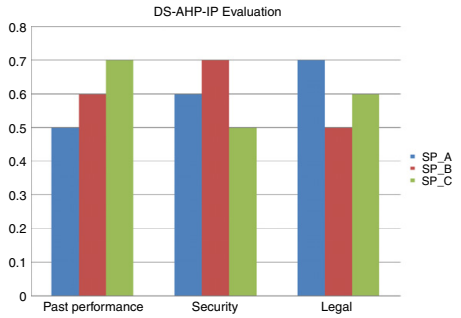


Fig. 11. IP-DS-AHP assessment of three SPs.

9. Concluding remarks

We present five fundamental challenges for wide adoption of cloud computing: service life cycle optimization, dependable sociability, adaptive self-preservation, multi-cloud architectures, and market and legislative issues. We believe that addressing these concerns as a whole is key to boost the delivery of advanced services. Our approach is two-fold, we perform fundamental research in a wide range of areas spanning from VM management and programming languages to business models and IT law to address these challenges, and we incorporate our findings in the development of the OPTIMIS Toolkit.

The focus of the toolkit is on cloud service and infrastructure optimization throughout the service life cycle: construction, deployment, and operation of services. All management actions in the toolkit are harmonized by overarching policies that consider trust and risk assessment to comply with economical and ecological objectives without compromising operational efficiencies. Assessing risk of economical and ecological parameters is a unique, albeit challenging, goal. Governance processes and policies are defined to harmonize management activities throughout the service life cycle.

The purpose of self-service tools is to enable developers to enhance services with non-functional requirements regarding allocation of data and VMs, as well as aspects related to elasticity, energy consumption, risk, cost, and trust. The OPTIMIS Toolkit incorporates risk aspects in all phases of the service life cycle and uses trust assessment tools to improve decision making in the matching of SPs and IPs. The ecological impact of service provisioning is integrated in all relevant decision making. The toolkit also ensures that the desired levels of risk, trust, or eco-efficiency are balanced against cost, to avoid solutions that are unacceptable from an economical perspective. The OPTIMIS tools are aimed to enable SPs and IPs to perform monitoring and automated management of services and infrastructures, so as to compare different alternative configurations in terms of business efficiency. Notably, mechanisms required to design policies that fulfill legislative and regulatory constraints are also taken incorporated in the toolkit, e.g., to address adoption challenges from regulatory and standards compliance requirements such as privacy and data protection.

Our goal is also to enable and simplify the creation of a variety of provisioning models for cloud computing, including cloud bursting, multi-cloud provisioning, and federation of clouds. Provisioning on multi-clouds architectures and federated cloud providers facilitates novel and complex composition of clouds that

considerably extend the limited support for utilizing resources from multiple providers in a transparent, interoperable, and architecture independent fashion.

Acknowledgments

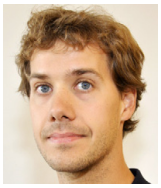
We acknowledge the anonymous reviewers for their insightful feedback. Financial support for this work is provided by the European Commission's Seventh Framework Programme ([FP7/2001-2013]) under grant agreement number 257115, OPTIMIS.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: a Berkeley view of cloud computing, in: Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2009. Technical Report No. UCB/EECS-2009-28.
[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.
[3] Luis Rodero-Merino, Luis M. Vaquero, Victor Gil, Fermín Galán, Javier Fontán, Rubén S. Montero, Ignacio M. Llorente, From infrastructure delivery to service management in clouds, Future Generation Computer Systems 26 (8) (2010) 1226–1240.
[4] L. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, SIGCOMM Computer Communications Review 39 (1) (2008) 50–55.
[5] Google. Google App Engine, Visited May 2011, code.google.com/appengine.
[6] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[7] C. Vecchiola, X. Chu, R. Buyya, Aneka: a software platform for. NET-based cloud computing, in: High Speed and Large Scale Scientific Computing, IOS Press, 2009, pp. 267–295.
[8] Microsoft. Windows Azure platform. Visited May 2011, www.microsoft.com/windowsazure.
[9] Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk, Zsolt N'emeth, An approach for virtual appliance distribution for service deployment, Future Generation Computer Systems 27 (3) (2011) 280–289.
[10] J. Li, O. Wäldrich, W. Ziegler, Towards SLA-based software licenses and license management in grid computing, in: Proceedings of the CoreGRID Symposium, Springer Verlag, 2008, pp. 139–152.
[11] D. Jordan, J. Evdemon (chairs), Web services business process execution language version 2.0, 2007. Visited June 2010, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.
[12] K. Keahey, T. Freeman, Contextualization: providing one-click virtual clusters, in: Fourth IEEE International Conference on eScience, IEEE, 2008, pp. 301–308.
[13] K. Keahey, M. Tsugawa, A. Matsunaga, J. Fortes, Sky computing, IEEE Internet Computing 13 (5) (2009) 43–51.
[14] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, Web services agreement specification (WS-agreement). May 2011, www.ogf.org/documents/GFD.107.pdf, Visited.
[15] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, R. Yahyapour, Establishing and monitoring SLAs in complex service based systems, in: Proceedings of the 2009 IEEE International Conference on Web Services, IEEE Computer Society, 2009, pp. 783–790.
[16] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F. Galán, The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 1–11.
[17] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, R.P. Doyle, Managing energy and server resources in hosting centers, ACM SIGOPS Operating Systems Review 35 (5) (2001) 103–116.
[18] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI'08, ACM, 2008, pp. 337–350.
[19] B.K. Alunkal, I. Veljkovic, G.V. Laszewski, K. Amin, Reputation-based grid resource selection, in: Workshop on Adaptive Grid Middleware, IEEE, 2003.
[20] K. Djemame, I. Gourlay, J. Padgett, G. Birkenheuer, M. Hovestadt, O. Kao, K. Voss, Introducing risk management into the grid, in: 2nd IEEE International Conference on e-Science and Grid Computing, e-Science'06, IEEE, 2006.
[21] M. Buco, R. Chang, L. Luan, C. Ward, J. Wolf, P. Yu, Utility computing SLA management based upon business objectives, IBM Systems Journal 43 (1) (2004) 159–178.
[22] D. Gilat, A. Landau, A. Sela, Autonomic self-optimization according to business objectives, in: 1st International Conference on Autonomic Computing, ICAC 2004, IEEE, 2004, pp. 206–213.
[23] A. McCloskey, B. Simmons, H. Lutfiyya, Policy-based dynamic provisioning in data centers based on slas, business rules and business objectives, in: IEEE/IFIP Network Operations and Management Symposium, NOMS'08, IEEE, 2008, pp. 903–906.
[24] Tian Chunqi, Baijian Yang, R2Trust, a reputation and risk based trust management framework for large-scale, fully decentralized overlay networks, Future Generation Computer Systems (March) (2011).
[25] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems 43 (2007) 618–644.
[26] Yining Liu, Keqiu Li, Yingwei Jin, Yong Zhang, Wenyu Qu, A novel reputation computation model based on subjective logic for mobile ad hoc networks, Future Generation Computer Systems 27 (5) (2011) 547–554.
[27] K. Djemame, I. Gourlay, J. Padgett, K. Voss, O. Kao, Risk Management in Grids, in: R. Buyya, K. Bubendorfer (Eds.), Market-Oriented Grid and Utility Computing, Wiley, 2009, pp. 335–353.
[28] L. Liu, H. Wang, X. Liu, X. Jin, W.B. He, Q.B. Wang, Y. Chen, GreenCloud: a new architecture for green data center, in: Proceedings of the 6th International Conference on Autonomic Computing, ACM, 2009, pp. 29–38.
[29] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, N. Gautam, Managing server energy and operational costs in hosting centers, ACM SIGMETRICS Performance Evaluation Review 33 (1) (2005) 303–314.
[30] J. Kephart, H. Chan, R. Das, D. Levine, G. Tesauro, F. Rawson, C. Lefurgy, Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs, in: 4th International Conference on Autonomic Computing, ICAC 2007, IEEE, 2007.
[31] Dang Minh Quan, Federico Mezza, Domenico Sannenli, Raffaele Giafreda, T-Alloc: a practical energy efficient resource allocation algorithm for traditional data centers, Future Generation Computer Systems (May) (2011).
[32] Damien Borgetto, Henri Casanova, Georges Da Costa, Jean-Marc Pierson, Energy-aware service allocation, Future Generation Computer Systems (May) (2011).
[33] P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, Adaptive control of virtualized resources in utility computing environments, ACM SIGOPS Operating Systems Review 41 (3) (2007) 289–302.
[34] Y. Song, Y. Sun, H. Wang, X. Song, An adaptive resource flowing scheme amongst VMs in a VM-based utility computing, in: 7th IEEE International Conference on Computer and Information Technology, CIT 2007, IEEE, 2007, pp. 1053–1058.
[35] J. Kephart, D. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50.
[36] Tiago C. Ferreto, Marco A.S. Netto, Rodrigo N. Calheiros, Csar A.F. De Rose, Server consolidation with migration control for virtualized data centers, Future Generation Computer Systems (May) (2011).
[37] J. Tordsson, R.S. Montero, R.M. Vozmediano, I.M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, 2010, in press (doi:10.1016/j.future.2011.07.003).
[38] W. Fellows, IT is cloud: to infrastructure and beyond, The 451 Group – CloudScape (June) (2010).
[39] H. Grant, T. Finlayson, Cloud computing & data protection. March 2009. Visited May 2011, http://www.twobirds.com/English/News/Articles/Pages/Cloud_Computing_Data_Protection_030609.Aspx.
[40] Art. 29 data protection working party. February 2010, http://ec.europa.eu/justice/policies/privacy/workinggroup/index_en.htm, Visited May 2011.
[41] P. Hustinx, Data protection and cloud computing under EU law, in: Third European Cyber Security Awareness Day, BSA, European Parliament, April 2010, pp. 1–7.
[42] The challenge of energy efficiency through information and communication technologies, February 2009. Parliament resolution of 4 February 2009. visited July 2010, http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+TA+P6-TA-2009-0044+0+DOC+XML+V0//EN.
[43] Computer OECD committee for information and communications policy. Recommendation of the council on information and communication technologies and the environment (OECD guidelines) — C(2010)61, April 2010. Visited May 2011, http://webnet.oecd.org/oecdacts/Instruments/ShowInstrumentView.aspx?InstrumentID=259.
[44] Y. Poullet, J. Van Gyseghem, J. Gerard, C. Gayrel, J. Moiny, Cloud Computing and its implications on data protection, chapter discussion paper prepared by the Research Centre on IT and Law — CRID, page 8. Council of Europe, March 2010. Visited May 2011, www.coe.int/t/dghl/cooperation/economiccrime/cybercrime/Documents/Reports-Presentations/2079_reps_IF10_yvespoullet1b.pdf.
[45] DMTF. Open Virtualization Format (OVF). Visited May 2011, http://www.dmtf.org/standards/ovf.
[46] A. Ali-Eldin, J. Tordsson, E. Elmroth, An adaptive hybrid elasticity controller for cloud infrastructures, 2011 (submitted for publication).
[47] G. Urdaneta, G. Pierre, M. van Steen, Wikipedia workload analysis for decentralized hosting, Elsevier Computer Networks 53 (11) (2009) 1830–1845.
[48] Z. Hua, B. Gong, X. Xu, A DS-AHP approach for multi-attribute decision making problem with incomplete information, Journal of Expert Systems with Applications 34 (2008) 2221–2227.

Ana Juan Ferrer has an Engineering Degree in Computer Software from Universitat Autònoma de Barcelona (1998). In Atos Origin since 2006, in Atos Research and Innovation, currently she is Head of Lab of the Service Engineering & IT Platforms Lab, group that focus its research in Cloud Computing, Service Engineering and Green IT. From June 2010 she manages the OPTIMIS project, investigating platforms and architectures for escalable and trustful Cloud services platforms. She also participates in NUBA, focusing in research on IaaS Cloud federation models. In the past, she has worked in NEXOF-RA, definition of the NESSI Open Framework Roadmap; and BEinGRID, industrial application of Cloud and Grid, and Crosswork project. Before Atos Origin she has wide experience as consultant and software architect in the Internet environment and e-business. Ana currently leads the Green IT working group in INES (Spanish S&S initiative), and participates in the ATOS Origin Scientific Community.

Francisco Hernández is an Assistant Professor in grid and cloud computing in the Department of Computing Science at Umeå University, Sweden. His research interests are in the areas of distributed systems and high performance computing. He holds a Ph.D. in Computer Science from the University of Alabama, Birmingham. He received the Graduate School Dean's Award for excellence in graduate studies at the Ph.D. level for the year 2006. Francisco currently serves as the co-chair of the Virtualized Service Platform Collaboration Workgroup (VSP CWG). He also holds a BS in Systems Engineering, from Universidad Francisco Marroquín, Guatemala.



Johan Tordsson is an Assistant Professor in Computer Science at Umeå University. Tordsson's research interests include several topics related to autonomic resource management for grid and cloud computing infrastructures, e.g., scheduling, elasticity, and quality of service. Academic background includes M.Sc., Lic.Eng, and Ph.D. Degrees from Umeå University and a postdoctoral visit at Universidad Complutense de Madrid. His current research activities include being the lead architect for the OPTIMIS Project (FP7 IP).

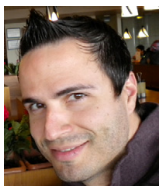
Erik Elmroth is a Professor and the Head of the Department of Computing Science at Umeå University. He is leading the Umeå University research on grid and cloud computing, including the group's participation in eSSENCE, the three EU FP7 IP-projects RESERVOIR, OPTIMIS, and VISION Cloud, and the UMIT research laboratory. Previous appointment highlights include an year at the NERSC, the Lawrence Berkeley National Laboratory, a semester at the Massachusetts Institute of Technology (MIT) as well as several years for Swedish and international research councils and governmental strategy bodies.

Ahmed Ali-Eldin is a Ph.D. student in the Cloud and Grid computing research group in Umeå University since October, 2010. He obtained his M.Sc. degree in Communication and Information Technology from Nile University, Egypt while working as a research assistant there (2008–2010). Ali-Eldin also holds a B.Sc. in Computer and Systems Engineering from Ain-Shams university, Egypt. His current research interest is self-* for clouds.

Csilla Zsigri is an economist with international work experience. Currently, she is the EU Business Development & Program Manager at The 451 Group (technology-industry analyst company). Csilla joined The 451 Group to extend business activities in Europe with special regard to EU research activities. Csilla also covers EU research trends, project outcomes and commercial opportunities for the 451 EURO service.

Previously, Csilla worked as a business consultant for Atos Origin's Research and Innovation unit in Barcelona, coordinating business and marketing activities in numerous European ICT Research and Technology Development projects. Before that, she did business development, consultancy work and in-company trainings for ConAction, a Hungarian management consulting company. Her focus was on seeking new business opportunities for the firm, as well as leading projects from launching new services on the local market to setting up new companies abroad. Csilla also has experience working at Nokia, being involved in a turnkey project for Vodafone, where she had to deal with key market players from the telecommunications industry.

Csilla holds an Economist M.Sc. degree specialized in Business and Management Consulting.



Raül Sirvent has a Ph.D. Degree in Computer Science at the Computer Architecture Department (UPC, 2009). He has been involved in research activities at the European Center of Parallelism of Barcelona (CEPBA) from 2002 to 2005. Since 2005, he has been holding a permanent position at the Barcelona Supercomputing Center inside the Computer Sciences department (Grid Computing and Clusters). His main research interests are related to high performance computing, grid and cloud programming models and tools, automatic workflow generation, fault tolerance mechanisms and the use of semantics for scheduling. He has been involved in FP6 projects CoreGRID, BEinGRID, and BREIN, the Spanish initiative "Red Temática para la Coordinación de Actividades Middleware en Grid" and one of the founders of the GRID superscalar project. Currently he is working in the Spanish project NUBA which focuses on the federation of cloud infrastructures, and the FP7 project OPTIMIS, which aims at optimizing cloud services taking trust, risk, eco-efficiency, cost and legal issues into account.

Jordi Guitart received the MS and Ph.D. Degrees in Computer Science at the Technical University of Catalonia (UPC), in 1999 and 2005, respectively. Currently, he is an associate professor at the Computer Architecture Department of the UPC and an associate researcher at Barcelona Supercomputing Center (B.SC.) within the Autonomic Systems and eBusiness Platforms research line. His research interests are oriented toward innovative resource management approaches for modern distributed computing systems. He is involved in a number of European projects.

Rosa M. Badia has a Ph.D. in Computer Science (1994) from the Technical University of Catalonia (UPC). She is a scientific researcher from the Consejo Superior de Investigaciones Científicas (CSIC) and manager of the Grid Computing and Cluster research group at the Barcelona Supercomputing Center (B.SC.). She was involved in teaching and research activities at the UPC from 1989 to 2008, where she has been an Associate Professor since 1997. From 1999 to 2005, she was involved in research and development activities at the European Center of Parallelism of Barcelona (CEPBA). Her current research interests are performance prediction and modeling of MPI programs and programming models for complex platforms (from multicore to the grid/cloud). She has published more than 100 papers in international conferences and journals on the topics of her research. She has participated in several European projects and is currently participating in projects NUBA (at Spanish level), TERAFLUX, TEXT, SIENA, OPTIMIS, VENUS-C and ScalaLife and is a member of HiPEAC2 NoE.

Karim Djemame (Co-Investigator) was awarded a Ph.D. at the University of Glasgow, UK, in 1999, and is currently holding a Senior Lecturer position at the School of Computing, University of Leeds. He sits on a number of international programme committees for grid/cloud middleware, computer networks and performance evaluation. He was the investigator of various e-Science/grid projects including DAME, BROADEN, and AssesGrid. His main research areas focus on grid/cloud computing, including system architectures, resource management, and risk assessment. Dr. Djemame is a member of the IEEE.

Wolfgang Ziegler is the head of the Grid and Cloud Middleware Research Group of the Department of Bioinformatics. His research areas are grid and cloud computing, resource management and scheduling, management of virtual organizations, and service oriented architectures. He has been the working group co-chair of the Open Grid Forum for more than 10 years. Currently, he is the OGF area director for applications and co-chairs the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG), which develops the OGF standard WS-Agreement and WS-Agreement Negotiation. He is participating in several national and European grid and cloud projects as work-package leader, where SLAs and license management in distributed computing environments play a major role, e.g. D-Grid, PHOSPHORUS, SmartLM, OPTIMIS.

Theo Dimitrakos

Srijith K. Nair is a Senior Security Researcher at BT Innovate & Design, the R&D part of BT where he looks at fundamental security challenges facing enterprise level infrastructures, including, but not limited to security aspects of cloud based services. He has experience in managing cross organizational teams, having served as a work package leader in European Union funded ICT project OPTIMIS. He was also part of multiple expert groups in the security industry that, among others, provide advice to the European Network and the Information Security Agency (ENISA). He has a Ph.D. in Computer Science from Vrije Universiteit, Amsterdam and has published several peer-reviewed papers in international journals, conferences and workshops and has also served on the program committee of several international conferences and journals. He is a member of the ACM and the IEEE.

George Kousiouris received his diploma in Electrical and Computer Engineering from the University of Patras, Greece in 2005. He is currently pursuing his Ph.D. in grid and cloud computing at the Telecommunications Laboratory of the Dept. of Electrical and Computer Engineering of the National Technical University of Athens and is a researcher for the Institute of Communication and Computer Systems (ICCS). He has participated in the EU funded projects BEinGRID IRMOS and OPTIMIS and the national project GRID-APP. In the past, he has worked for private telecommunications companies. His interests are mainly computational intelligence, optimization, computer networks and web services.

Kleopatra Konstanteli received her diploma in Electrical and Computer Engineering in 2004 from the National Technical University of Athens (NTUA). In 2007, she received a Master's Degree in Techno-economical Systems from the NTUA in cooperation with the University of Athens and the University of Piraeus. She is currently pursuing her doctoral-level research in Computer Science, and at the same time, working as a research associate in the Telecommunications Laboratory of Electrical and Computer Engineering of NTUA and participating in EU funded projects. Her research interests are mainly focused on the field of distributed computing.

Theodora Varvarigou received the B. Tech Degree from the National Technical University of Athens, Athens, Greece in 1988, the MS Degrees in Electrical Engineering (1989) and Computer Science (1991) from Stanford University, Stanford, California in 1989 and the Ph.D. Degree from Stanford University as well in 1991. She worked at AT&T Bell Labs, Holmdel, New Jersey between 1991 and 1995. Between 1995 and 1997, she worked as an Assistant Professor at the Technical University of Crete, Chania, Greece. In 1997, she was elected as an Assistant Professor, and since 2007, she has been a Professor at the National Technical University of Athens, and the Director of the Postgraduate Course "Engineering Economics Systems". Prof. Varvarigou has great experience in the area of semantic web technologies, scheduling over distributed platforms, embedded systems and grid computing. She has published more than 150 papers in leading journals and conferences in this area.

Benoit Hudzia is a Senior Researcher at SAP Research, CEC Belfast. He is leading the UK funded project Virtex on virtualization technologies, participates in the EU-funded cloud computing project, Reservoir, and is involved in the SAP internet of the services framework research program. He received a Ph.D. from the University College, Dublin in the field of parallel and distributed computing, focussing on P2P and grid systems. During his studies in Electronics and Computer Science Engineering at the University of Paris 6 (M.Sc.) and the engineering school EFREI (Paris) (Meng), his main emphasis was on distributed systems and parallelism for enterprise applications.

Alexander Kipp holds a Diploma in Computer Science. Since 2006, he has been employed at the High Performance Computing Center (HLRS) at the University of Stuttgart, working on cloud-/VO-/web service-related research projects on both national and international levels. He is the deputy head of the Intelligent Service Infrastructures department at HLRS, which focuses on realizing next generation cross domain service infrastructures. In this context, he is responsible for the Green-IT activities of HLRS. His research interest lies in the areas of adaptive and energy aware service infrastructures, in particular, in the area of service virtualization and corresponding technologies.

Stefan Wesner holds a Ph.D. in Mechanical Engineering from the University of Stuttgart and a diploma in Electronic Engineering from the University of Saarland. He is the Managing Director of the High Performance Computing Center, Stuttgart and heads the Applications and Visualization Department. His current research interests are automated management, SLA-based service provision for distributed environments and new paradigms for parallel computing. He was one of the first involved in the research on business-oriented grids and the application of service-level agreements for grids. He was the technical coordinator of the Akogrimo and BREIN projects and is currently involved in several research projects in the fields of cloud and high performance computing.

Marcelo Corrales holds an LL.M degree, and is a lawyer admitted to the Paraguayan Bar Association since 2004. In 2007, he got a Master's Degree in Law and Information Technology and, in 2009, a Master's Degree in European Intellectual Property Law from Stockholm University. Since October 2007, he has been a research associate at the Institute for Legal Informatics (IRI). His research interests include intellectual property rights and privacy issues in the realm of information law.

Nikolaus Forgó studied Law, Philosophy and Linguistics in Vienna and Paris. In 1997, he got his Dr. iur. (Dissertation in legal theory). Between 1990 and 2000, he worked as an Assistant Professor at the University of Vienna (Austria). Since 2000, he has been a full-time Professor for Legal Informatics and IT-Law at the University of Hanover; since 2007, the co-head of the Institute for Legal Informatics. Publications, teaching and consulting experience in all fields of IT-law, legal informatics, civil law, legal history and legal theory on national and international levels.

Tabassum Sharif completed his B. Eng. in Electronic and Electrical Engineering at the School of Electrical and Electronic Engineering with the Corp of Royal Electrical and Mechanical Engineers. He has spent almost 8 years within the military specializing in telecommunications and various other communication projects. He has previously worked with organizations within the financial services industry, insurance industry and prepayment industry before coming to the ISP industry where he is currently employed as the Operations Manager. He brings a wealth of experience in translating theoretical ideologies and best practices into real world environments.

Craig Sheridan holds a B.Sc.(Hons) in Network Computing and has worked for Flexiant since inception in various roles including project management and systems administration and has been a Support Manager and Systems Administrator for XCalibre Communications for the past 5 years and has various IT related certifications. He has a wide range of expertise in cloud technology. Prior to this, he worked for 8 years for Motorola as a Radio Frequencies Analyser.

Paper VII

Infrastructure Federation through Virtualized Delegation of Resources and Services[7]

Georg Birkenheuer[1], André Brinkmann[1], Mikael Högqvist[2], Alexander Papaspyrou[3],
Bernhard Schott[4], Dietmar Sommerfeld[5], Wolfgang Ziegler[6]

[1]Paderborn Center for Parallel Computing, University of Paderborn, Paderborn, Germany
{birke, andre.brinkmann}@uni-paderborn.de

[2]Zuse Institute Berlin, Berlin, Germany
hoegqvist@zib.de

[3]Technische Universität Dortmund, Dortmund, Germany
alexander.papaspyrou@tu-dortmund.de

[4]Platform Computing GmbH, Ratingen, Germany
bschott@platform.com

[5]Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Göttingen, Germany
dsommer@gwdg.de

[6]Fraunhofer Institute SCAI, Schloss Birlinghoven, 53754 Sankt-Augustin, Germany
wolfgang.ziegler@scai.fraunhofer.de

Abstract: Infrastructure federation is becoming an increasingly important issue for modern Distributed Computing Infrastructures (DCIs): Dynamic elasticity of quasi-static Grid environments, incorporation of special-purpose resources into commoditized Cloud infrastructures, cross-community collaboration for increasingly diverging areas of modern e-Science, and Cloud Bursting pose major challenges on the technical level for many resource and middleware providers. Especially with respect to increasing costs of operating data centers, the intelligent yet automated and secure sharing of resources is a key factor for success. With the D-Grid Scheduler Interoperability (DGSI) project within the German D-Grid Initiative, we provide a strategic technology for the automatically negotiated, SLA-secured, dynamically provisioned federation of resources and services for Grid-and Cloud-type infrastructures. This goal is achieved by complementing current DCI schedulers with the ability to federate infrastructure for the temporary leasing of resources and rechanneling of workloads. In this work, we describe the overall architecture and SLA-

[7]By permission of Springer Verlag

secured negotiation protocols within DGSI and depict an advanced mechanism for resource delegation through means of dynamically provisioned, virtualized middleware. Through this methodology, we provide the technological foundation for intelligent capacity planning and workload management in a cross-infrastructure fashion.

Keywords: Gridcomputing, Cloudcomputing, Resource management, Negotiation, Virtualization, Middleware, SLAs

J Grid Computing (2011) 9:355–377
DOI 10.1007/s10723-011-9192-1

Infrastructure Federation Through Virtualized Delegation of Resources and Services

DGSI: Adding Interoperability to DCI Meta Schedulers

Georg Birkenheuer · André Brinkmann · Mikael Högqvist · Alexander Papaspyrou · Bernhard Schott · Dietmar Sommerfeld · Wolfgang Ziegler

Received: 28 February 2010 / Accepted: 12 June 2011 / Published online: 7 July 2011

Abstract Infrastructure federation is becoming an increasingly important issue for modern Distributed Computing Infrastructures (DCIs): Dynamic elasticity of quasi-static Grid environments, incorporation of special-purpose resources into commoditized Cloud infrastructures, cross-community collaboration for increasingly diverging areas of modern e-Science, and Cloud Bursting pose major challenges on the technical level for many resource and middleware providers. Especially with respect to increasing costs of operating data centers, the intelligent yet automated and secure sharing of resources is a key factor for success. With the D-Grid Scheduler Interoperability (DGSI) project within the German D-Grid Initiative, we provide a strategic technology for the automatically negotiated, SLA-secured, dynamically provisioned federation of resources and services for Grid-and Cloud-type infrastructures. This goal is achieved by complementing current DCI schedulers with the ability to federate infrastructure for the temporary leasing of resources and rechanneling of workloads. In this work, we describe the overall architecture and SLA-secured negotiation protocols within DGSI and depict an advanced mechanism for resource delegation through means of dynamically provisioned, virtualized middleware. Through this methodology, we provide the technological foundation for intelligent capacity planning and workload management in a cross-infrastructure fashion.

This work is supported by the German Federal Ministry of Education and Research under project grant #01IG09009.

G. Birkenheuer · A. Brinkmann
Paderborn Center for Parallel Computing, University of Paderborn, Paderborn, Germany

G. Birkenheuer
e-mail: birke@uni-paderborn.de

A. Brinkmann
e-mail: andre.brinkmann@uni-paderborn.de

M. Högqvist
Zuse Institute Berlin, Berlin, Germany
e-mail: hoegqvist@zib.de

A. Papaspyrou ( )
Technische Universität Dortmund, Dortmund, Germany
e-mail: alexander.papaspyrou@tu-dortmund.de

B. Schott
Platform Computing GmbH, Ratingen, Germany
e-mail: bschott@platform.com

D. Sommerfeld
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Göttingen, Germany
e-mail: dsommer@gwdg.de

W. Ziegler
Fraunhofer Institute SCAI, Sankt Augustin, Germany
e-mail: Wolfgang.Ziegler@scai.fraunhofer.de

Keywords Grid computing · Cloud computing · Resource management · Negotiation · Virtualization · Middleware · SLAs

1 Introduction

Grid computing technology has reached a state of maturity where former enthusiastic users have become customers, and "nice-to-have" resources of (mostly) academic origin, usually occupied with cutting-edge research problems, start to provide value to projects driven by the needs of the very businesses they comprise.

A good example is climate research: Starting to explore Grid infrastructures as a new way to cope with tera-scale[1] data sets, the climate community now sees its portal-based gateways to resources as a tool to provide crucial information to members of parliament in the European member states and to decision makers from major reinsurance companies. Another case is provided by plasma technology consultancy: Within a business ecosystem of Small and Medium Enterprises (SME) in mechanical engineering, physicist modelers of plasma gas behavior, and consultancy bureaus, Grid technology is employed in a very similar way. Customers that build new machinery for making next-generation material science through plasma coating happen to give their blueprints to highly specialized consultants who—using Grid technology for highly compute-intensive simulations—answer questions regarding the feasibility of certain configurations by relying upon the subtle physical interaction models only specialist physicists can provide.

Overall, the usage of Grid beyond research is starting to plant its roots. Of course, this new customer base has different requirements for a Grid: Aspects such as resource reliability, performance guarantees, and data secrecy become more important than they have been in classic Grid use cases that stem from academia. The Intergovernmental Panel of Climate Change (IPCC), for example, demands results on a certain quality level, and plasma consultants have tight deadlines for their reports. As such, both projects start to look out for a more flexible approach to reliably integrate additional resources at times of demand.

Besides Grid computing, the more recent concept of Cloud Computing has reached the market, beginning to deliver market value and generating revenue in the large scale. The natural idea for the aforementioned use cases would thus be to embrace Infrastructure as a Service (IaaS) concepts for their natural purposes. Still, the precious compute and storage infrastructure and community-specific high level services from the Grid era (think intelligent special-purpose data management and highly customized portal applications) that are successfully used by a large community cannot be sacrificed. In addition, there is a strong need to keep the collaborative VO-based approach: Partners, albeit competing at the market, want to collaborate over the common platform. Discussions with the different stakeholders elicited the wish for convergence between the traditional Grid approach, with more or less static infrastructure environments and value-added community services, and the elastic yet "baseline services only"-enabled Cloud approach.

Through the use of open standards, the D-Grid Scheduler Interoperability (DGSI) project aims to make this convergence happen on the technology level. Initially started as an effort to make cross-VO collaboration between Service Grids possible and to connect researchers from different communities on the infrastructure level, the project extended its goals towards linking traditional Grid environments with modern elastic infrastructures. The support for auto-negotiated "leasing" of resources and delegation of workloads between different types of infrastructures opens new opportunities of cross-community and cross-infrastructure interoperation, and strict end-to-end Service Level Agreements ensure the applicability to business-driven Grid communities. This federation of different paradigms for resource exposure forms a key issue in the European resource space. Open standards, especially from organizational bodies that think into

[1] Terabytes of data per input set for a single task in a complex workflow.

both Grid and Cloud directions, build the technological foundation for this challenge. By adopting these standards for both academic and commercial environments, collating them according to the requirements of business-driven Grid infrastructures and contributing back to the standards organizations, DGSI provides a strategic technology platform for the future convergence of Grid and Cloud computing.

In this paper, we describe the DGSI architecture, protocols, and technologies and detail two approaches for DCI federation: In *resource delegation*, a meta scheduler can place a resource available within its own domain under the exclusive planning authority of another meta scheduler outside of his administrative control, for a previously negotiated time window. In activity delegation, one meta scheduler hands hands an activity and the management of its execution over to the management domain of another scheduler. Both approaches aim to work in a federated, cross-community, cross-infrastructure scenario.

The remainder of the paper is organized as follows: In Section 2, we give an overview of related work with respect to capacity planning in federated DCI infrastructures and show various projects employing negotiation and agreement mechanisms for workload management. In Section 3, we give an overview of the DGSI architecture, providing details on the two delegation approaches. In Section 4, we discuss the negotiation and agreement details with respect to the DGSI protocol and show the integration of open standards within the whole system. In Section 5, we show the details of the most advanced approach, namely virtualized resource delegation and discuss technological challenges in realization. In Section 6, we detail the information model used in DGSI. In Section 7, we evaluate our approach using a test environment. Finally, in Section 8, we conclude our work with prospects for future development.

2 Related Work

Federation aspects in general have been discussed mostly from a security perspective [11], on the level of networks [33], or for monitoring purposes [7]. Although standardization is considered a key case [15] and has been shown on an operational level [32], federation protocols for automatic, infrastructure-level, and SLA-secured provisioning of resources and services are yet to be shown.

This notion of federating Grid and Cloud infrastructures on the broker level touches two major topics addressed by research in the last decade: scheduling and planning federated DCI infrastructures and interoperation protocols for negotiation and agreement between independently acting brokers in the domain of resource management.

2.1 Capacity Planning in Modern DCI Environments

Automated workload scheduling in distributed computing infrastructure (DCI) systems is a well-covered research topic and stems from classic parallel machine scheduling problems. The federated nature of such systems, however, allows a much more efficient usage of resources due to the shared utilization by a large user community [5].

During the last years, a remarkable amount of effort has been put into workload delegation within federated DCI ecosystems, mostly within the broader context of Grid computing. One popular approach in this scenario is to assume centralized scheduling services [18]. For example, Ernemann et al. [13] show advantages of hierarchical scheduling in general by considering the Average Weighted Response (AWRT) objective. Furthermore, Kurowski et al. [25] identify multiple objectives for efficient job scheduling in Grids and propose a strategy based on prediction mechanisms and resource reservation. For decentralized environments, only few results that support the delegation of workload have been published. England and Weissman [12] give an estimation of load sharing costs and benefits relying on synthetic workloads only. Grimme et al. [20] analyze the prospects of collaborative job sharing and compare their results to the non-cooperative scenario of the same machines. Recent work of

Fölling et al. [16, 17] proposes a fuzzy-based, evolutionary-optimized exchange policy for a fully decentralized scenario, which shows robustness even in changing environments and automatically adapts to the current local load.

With the elasticity of IaaS-supported DCI environments, a new kind of flexibility challenges current scheduling approaches due to the inherent reconfigurability of machines and the resulting changes in scheduling responsibilities. Up to now, this aspect has only occasionally been discussed in research: Since the complexity of operating such systems in the large scale and the feasibility of provisioning and reconfiguring them on demand hampered the realization for production environments, discussion focused on rather low-level computer hardware. For example, Kota et al. [24] consider the problem of scheduling and mapping of tasks onto reconfigurable logic units for a given application introducing a concept of parameterized modules. Their approach is a typical example of scheduling in the context of reconfigurable hardware that involves varying sizes of available hardware. Subramaniyan et al. [39] transferred similar ideas to the HPC context and analyzed the dynamic scheduling of large-scale HPC applications in parallel reconfigurable computing environments. They assess the performance of several common HPC scheduling heuristics that can be used by an automated job management service to schedule application tasks on parallel reconfigurable systems. However, their approach is limited to a single HPC system and does not involve the interaction of multiple autonomous partners in a DCI environment.

Iosup et al. have introduced the concept of delegated matchmaking, where resources from remote Condor pools can be bound temporarily to the local environment [22]. The solution augments a hierarchy of Grid sites with peer-to-peer connections between sites on the same hierarchy level. The basic idea of the approach is to combine advantages of hierarchical and distributed approaches to manage interoperating Grids. The main aim of the work is to decrease the necessary administrative effort by delegating resources to jobs instead of moving jobs to resources.

All approaches, however, share a strong focus on algorithmic improvements and assume an appropriate delegation infrastructure to be in place.

2.2 Negotiation and Agreement for Resource Management

During the last couple of years, the interest in using negotiation during job submission increased constantly. We identify two main reasons for this:

1. With Grid and Cloud infrastructures becoming an accepted way to extend the local resources, job submission may include resources beyond the own administrative domain. As a consequence, Quality of Service (QoS) of a remote submission cannot be enforced through local policies but has to be agreed upon with the external service provider, i.e., an agent like a meta-scheduler acting on behalf of this service provider.
2. Standards for creating such agreements, namely GLUE [2] for a description of resources, JSDL [4] for describing the properties of a job, and WS-Agreement [3] for negotiating and creating the agreement on resource usage and the related QoS, have been adopted and thus become available.

This growing interest is reflected by the increasing number of projects and developments in different areas implementing negotiation for job submission. Two surveys in 2007 [30, 35] identified more than 10 projects by that time either already providing an implementation, working on an implementation or planning to implement negotiation in existing systems, e.g., VIOLA MetaScheduling Service, AssessGrid CCS, ASKALON, Community Scheduling Framework (CSF), AgentScape, CATNETS, Job Submission Service (JSS), GRMS, GridWay, eNanos, and Grid Superscalar.

More recently, a number of projects or developments have been launched or completed that address the negotiation and creation of Service Level Agreements for resource management at different levels:

PHOSPHORUS uses negotiation and Service Level Agreements to co-allocate computational resources and network resources for submissions of jobs, which require a dedicated QoS for both of these resources [31].

BREIN addresses outsourcing to increase the competitiveness of SMEs by enhancing the classical Grid solutions through integration of multi-agent and semantic web concepts into a dynamic, standards-based environment for eBusiness. The main emphasis has been on moving away from the Grid approach of handling individual resources, to a framework, allows providing and selling services which represent a combination of different resource types [10].

SmartLM focuses its research on the negotiation of the co-allocation of software licenses and Grid resources with respect to job submission and execution for license-protected applications in a Grid environment [37].

Bazaar is a tool aiming to make the process of SLA-based resource allocation manageable and transparent. The SLA layer is used to filter out appropriate resources for a job or to identify missing resources. Bazaar is implemented in the Central European EGEE-III region [8].

SORMA addresses job submission on a more abstract level. It is a research project investigating the practical application of computational and other related resources in a market infrastructure. A key aspect of this approach is the introduction of potential consumers to fitting resource providers, which is facilitated through asynchronous auctions, where "best" fits are identified and introduced [38].

SLA@SOI addresses the entire framework for SLAs in Service Oriented Infrastructures (SOI). The project focuses on research and development of the entire architecture for SLAs in SOIs including the provision of the overall technical framework and the business requirements [36].

Although the usage of negotiation and agreement for the management of workload has been adopted by many research efforts, the notion of employing such mechanisms for infrastructure federation has yet to be tackled.

3 Overview of the Federation Architecture of DGSI

Most DCI environments share the ability to efficiently distribute user workload to the resources available. This issue, usually generalized under the term *Meta Scheduling*, is already very diverse within a community: Both submitted jobs and available resources differ considerably, to the extent that coordination has to handle specialized knowledge about usage scenarios and infrastructure. This leads to very different, community-specific approaches for the development of Grid scheduling services.

The D-Grid Scheduler Interoperability (DGSI) provides a standards-based interoperability layer for scheduling and resource management services in DCI ecosystems. The DGSI solution allows the users of a community to distribute the workload among resources within the management domain of another community. It offers new perspectives for community collaboration, resource federation, and efficient utilization.

A general overview of the DGSI architecture is given in Fig. 1. For this approach, we foresee two scenarios to be realized within the framework: the delegation of activities and the delegation of resources.

3.1 Delegation of Activities

By means of DGSI, meta-schedulers are able to exchange activities, i.e., single or parallel jobs or workflows. In a first step, each meta-scheduler registers at a distributed registry and publishes its resources and execution capabilities. Now, another registered scheduler is able to query for the capabilities that are requested by an activity it wants to delegate. Furthermore, a session object with limited lifetime is generated, which provides a unique interface to a single or multiple activities and assures a thread safe access to them. This

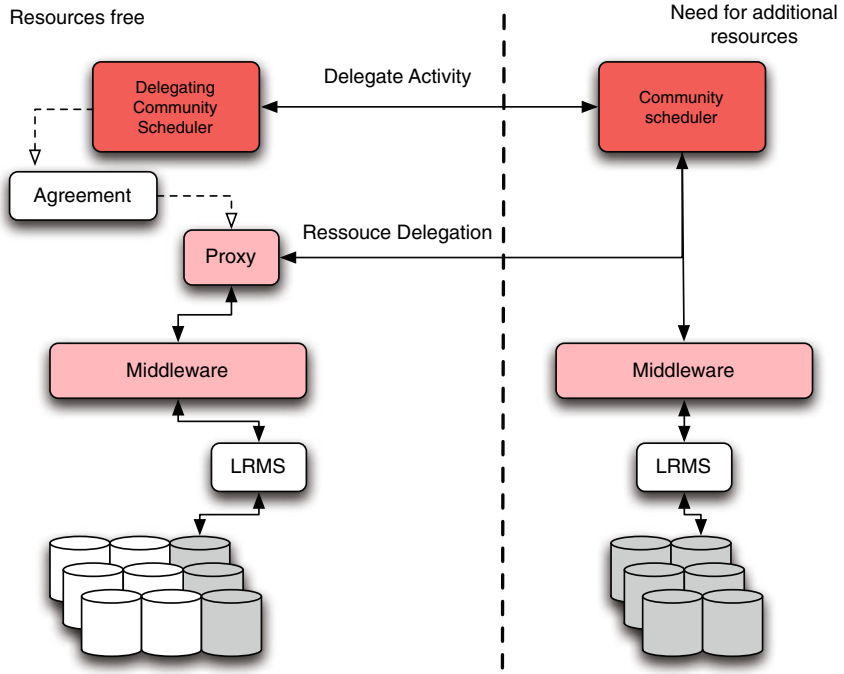


Fig. 1 Overall delegation architecture using the DGSI protocols. Meta schedulers from different domains (architectural, organizational, and technological) cooperate using activity and resource delegation

session object containing activities is offered to one or more meta-schedulers that match the demands. Every queried meta-scheduler is inspects the session and its containing activities independently and decides whether to accept or deny them. This phase could contain different negotiation protocols implementing different behavior (*Take it or leave it*, *Renegotiation by use of counter-offers* etc.). Eventually, one distinct meta-scheduler—either a local or a remote one—will execute the activity on its local resources.

3.2 Delegation of Resources

The method of resource delegation has become an interesting approach due to its advantages, like support of local requirements, e.g., special workflow scheduling tools or better control over the delegated resources. Resource delegation aims to

1. Encapsulation of the delegated resources,
2. Monitoring of the agreed terms of usage , and
3. Hiding of the security-relevant adaptations and of the delegation of rights.

The DGSI project has identified two distinct technologies for resource delegation.

The first realization is based on a proxy approach, which focuses on the management of delegated resources and is exclusively realized on the layer of the base middleware. The communication between Grid scheduler and middleware is directed over the proxy which enforces the compliance with the negotiated SLA. In this approach the provider keeps full control over the infrastructure. No modifications are required for the existing middleware and LRMS systems.

The second realization is based on virtualization. This approach allows the configuration of the base middleware and lightweight local RMS system in a virtual machine. The approach is flexible since the resources can be prepared for every middleware or LRMS. Access rights for the resources can be confi gured as part of the deployment.

While both delegation approaches are going to be implemented within the project, in this paper we will only focus on architectural aspects of the latter and detail the technical issues of the most advanced realization, namely the virtualization

approach. To this end, we first introduce the basic negotiation and agreement mechanisms and their technical rendering.

4 Protocol Definition for Negotiation-based Delegation of Services

The requirements for the architecture and the protocol language for delegation were discussed [21] in the Open Grid Forum's Grid Scheduling Architecture Research Group (GSA-RG) for some time. The results of this work, along with the results and experiences from a number of projects (see Section 2.2) strongly suggest two major design decisions for managing delegations:

1. An agreement on resource delegation can only be achieved by a negotiation process. The different administrative domains that make up a Grid or a Cloud infrastructure simply prohibit enforcement of remote resource usage.
2. Use of standards as far as possible to increase interoperability and usability.

The DGSI approach is thus based on three relevant standards to realize its delegation protocol:

- WS-Agreement to negotiate Service Level Agreements for using remote resources, for delegating activities to them or for temporary logical inclusion into the local resource pool.
- Job Submission Description Language (JSDL) to describe requirements for the delegation. Use case elicitation has shown that JSDL is suitable to describe the requirements for both resource and activity delegation (using only a subset of the language).
- GLUE to describe resources, which will be contributed for resource delegation.

The following sections present WS-Agreement and JSDL, the rationales for using them and describe their use and implementation in the DGSI approach. Although GLUE plays an important role in the protocol definition, it merely provides the information modeling part for describing resources. As such, it is described separately in Section 6.

4.1 WS-Agreement

WS-Agreement is a proposed recommendation of the Open Grid Forum defined by the Grid Resource Allocation Agreement Protocol working group (GRAAP-WG) [19]. The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of service providers and creating agreements based on templates and for monitoring agreement compliance at runtime. An agreement between a service consumer and a service provider specifies one or more service level objectives as expressions of the service consumer's requirements and the service providerÅŘs assurances on the availability of resources and/or service qualities. An agreement life cycle includes creation and termination and monitoring of agreement states. For example, an agreement may provide assurances on the bounds of service response time and service availability. Alternatively, it may provide assurances on the availability of minimum resources such as memory, CPU MIPS, storage, or a software license.

4.1.1 Rationale

We selected WS-Agreement to implement the basic protocol for resource delegation for the following reasons:

1. Delegation of resources or activities across autonomous communities can only be achieved through negotiation of the delegation between the requesting and the providing community Grid-scheduler.
2. For a negotiation it is essential that a protocol exists, which can be followed, and that the two negotiating parties have a common understanding of the objects the negotiation is about. Both can be achieved with WS-Agreement through its protocol and the use of agreed term languages (like JSDL) to describe the objects.
3. The negotiations should result in Service Level Agreements with binding character for both parties to deliver a reliable service to the end-user. Such SLAs are the outcome of successful negotiations with WS-Agreement.

4. The D-Grid infrastructure is currently being extended with a layer for the management of Service Level Agreements. This layer will be using WS-Agreement for the SLAs.

In addition to this, WS-Agreement enjoys wide adoption in the context of DCI environments because of two important properties.

WS-Agreement is standard. WS-Agreement is the only open standard specifying a language and a protocol for creating Service Level Agreements. Besides WS-Agreement only proprietary solutions are available, most of them developed in and for a certain ecosystem and no longer maintained or further developed, e.g,. the NextGRID SLA [27], or available under a commercial license only, like IBM's WSLA [41]. WS-Agreement uses the Web Services Resource Framework (WSRF) [42], an open framework for modeling and accessing stateful resources using Web services, which has been published as a standard by the OASIS consortium. The current version of UNICORE and the most popular edition of Globus Toolkit employ WSRF as a key technology; thus, basic interoperability for accessing stateful resources is already provided by the middleware.

WS-Agreement is extensible. The intrinsic extensibility of WS-Agreement is an important feature when using it as the general technology for creating SLAs in the heterogeneous environment of D-Grid comprising 20+ communities with different resources. Since the specification is completely neutral with respect to specific term languages to express Service Description Terms (SDT) and Service Level Objectives (SLO), DGSI can select the best suited description languages to be used for resource delegation and plug them into WS-Agreement. This mechanism allows providing a common approach for creating agreements on delegation while supporting a wide range of heterogeneous resources and activities. While the current specification of WS-Agreement only provides a basic, single round mechanism for negotiating an agreement, developments are under way at OGF to extend the negotiation capabilities towards multi round negotiations for creating an SLA and re-negotiation of SLAs already in force. A first implementation of WS-Agreement with extensions for negotiations has been realised in the SmartLM project [34]. In fact, we already started integrating the draft for the WS-Agreement Negotiation specification for the second phase of the DGSI project.

4.1.2 Implementation

As mentioned before, DGSI will rely on the SLA management layer of D-Grid, which will base on, extend, and port existing implementations of WS-Agreement to benefit from previous experiences, e.g., described in [6]. One of the most complete and advanced implementations of the WS-Agreement specification is WS-Agreement for Java (WSAG4J) [40], which provides an easy to use framework to create, monitor, and terminate service level agreements based on the WS-Agreement specification. To accomplish this goal, the framework implements the basic features of the WS-Agreement protocol and uses a number of standards in conjunction with WS-Agreement. Hence, WSAG4J provides a complete development framework for SLA-based services. DGSI uses features such as the WS-Agreement *AgreementFactory* port type and the WS-Agreement *AgreementState* port type. It makes use of digital signatures to enforce message integrity (WS-Security) and also provides the listing of existing agreements via WS-Service Groups.

Before executing an agreement, the agreement offers are validated based on template creation constraints. The validation is also available during the negotiation process. Limiting the negotiation space with creation constraints defined by the agreement initiator and/or the agreement responder reduces the complexity of a negotiation and allows to achieve more rapid convergence of a negotiation towards an agreement. As described in the WS-Agreement specification, an agreement factory of the resource provider exposes an operation for creating an agreement out of an initial set of terms and returns an Endpoint Reference (EPR) to an Agreement service. The agreement factory of the resource provider also exposes resource properties like templates—representations of acceptable offers for the creation of an agreement. In DGSI these templates

contain either the description of resources accepting activity delegation or the descriptions of resources a provider is willing to temporarily delegate. To create an agreement, a client—in DGSI a DCI scheduler—makes an agreement offer based on a previously retrieved template. Once the two parties—the scheduler requesting the delegation and the scheduler offering resources for delegation—agree on the terms of the template, the agreement may be created.

After the creation of the agreement, it will be monitored to verify whether the agreement is fulfilled or violated. For monitoring the agreement, several states are defined in the WS-Agreement specification (see Fig. 2):

- *Agreement States* corresponding to the agreement as a whole,
- *Service Term States* reflecting the states related to the service terms, and
- *Guarantee States* indicating the states of the guarantees given for the service terms.

WSAG4J implements these states and provides the interfaces to access them for setting and retrieving. However, it should be noted that the information accessible through Service Term States and Guarantee States is not provided by WS-Agreement internal mechanisms but must be gathered by external monitoring services (Fig. 3). For this purpose the DGSI approach is relying on external information and monitoring services available in D-Grid.

4.2 Job Submission Description Language

The description of resources on the negotiation level can be realised with JSDL. Basically, we have one resource-specific JSDL profile. It is used to describe (a) resources that a provider is willing to delegate to another scheduler domain and (b) resources requested by a scheduler aiming to temporarily gain control over remote resources.

4.2.1 Rationale

JSDL was originally developed to describe resource requirements for job requests. As such, JSDL contains several attributes, which are also applicable for resource delegation requirements. Moreover, JSDL can also be extended using profiles that address additional attributes required for resource delegation.

4.2.2 Implementation

In the first phase of DGSI, JSDL is used as it is and will be extended as necessary in the second phase to accommodate additional require-

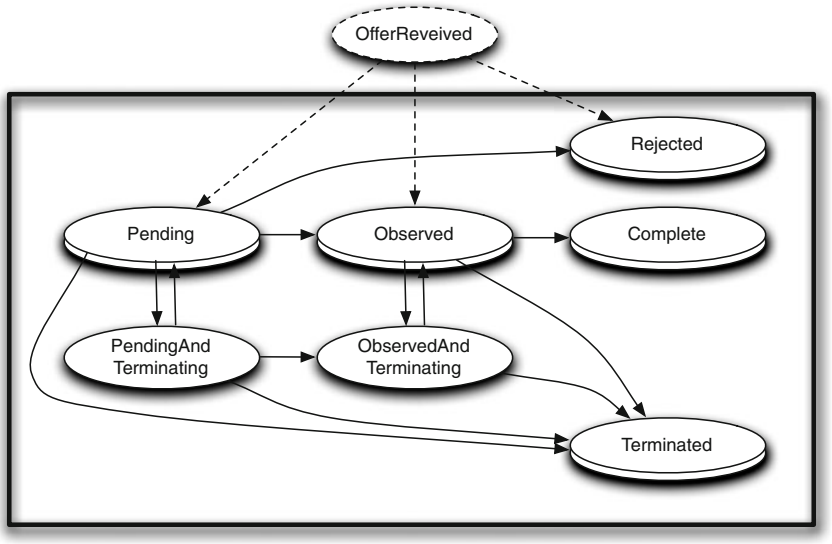
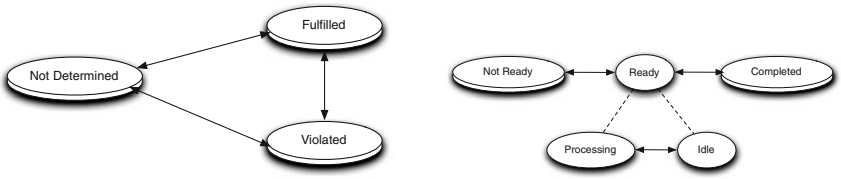


Fig. 2 Agreement States as defined in WS-Agreement, including transitions to each other. Note that the *OfferReceived* state is supplemental to the model



(a) Service TermStates. (b) Guarantee Term States.

Fig. 3 Transition diagram for the WS-Agreement Service and Guarantee Term States

ments. A potential extension for resource delegation is to support JSDL with a GLUE 2.0 extension. This would allow Grid-schedulers to craft delegation requests containing requirements for, e.g., the application environment and access policies.

4.3 Delegation Protocol Steps

To depict the integration of the aforementioned standards within the DGSI protocol stack, we show a coarse overview of the necessary steps to perform a delegation (see Fig. 4):

1. The requesting scheduler fetches available SLA templates from the providing scheduler. These can either describe abstract resources (depicting the general characteristics, that is), which basically serve as job slots for activity delegation, or concrete machine specifications for resource delegation.
2. The providing scheduler returns a set of templates describing services it can offer with respect to the current situation. These can now

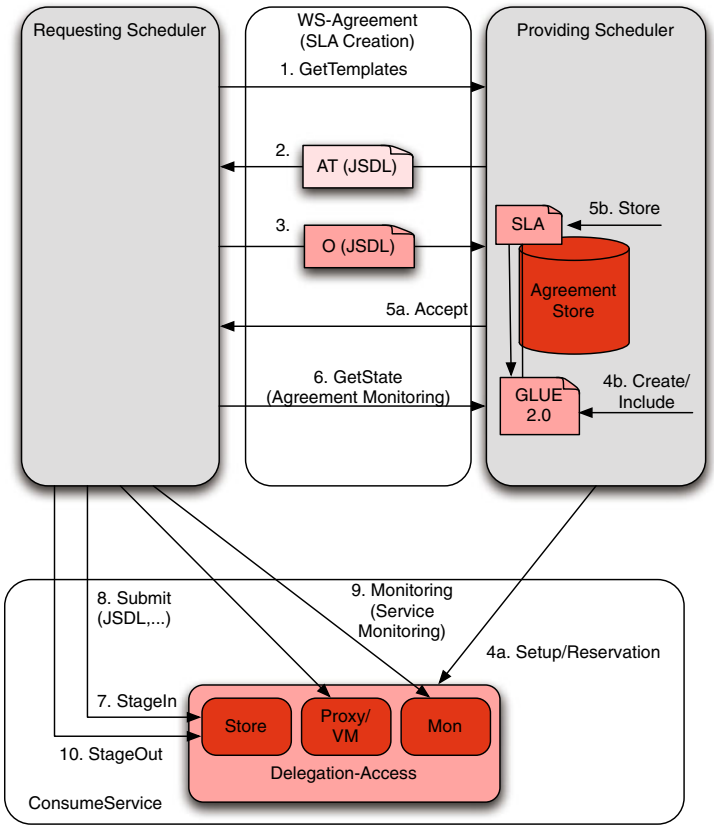


Fig. 4 Ten steps for negotiating a delegation within the DGSI protocol stack. While activity delegation only requires six steps (up to the *GetState* call for agreement monitoring), resource delegation is brought to completion with the tenth step. Note, however, that—from step six on—each step is specific to the concrete usage of the delegated resource (e.g., a single job submission)

be concretized with additional requirements from the requesting scheduler's side.
3. The requesting scheduler offers the concretized template (depicting the aspired SLA) to the providing scheduler.
4. The providing scheduler checks the general feasibility and adherence to creation constraints of the SLA and, if correct, creates a corresponding reservation for the delegation (a); this step can also be performed asynchronously. In case of resource delegation, the providing scheduler additionally sets up a GLUE specification of the delegated resource along with the stored agreement.
5. Regardless whether the previous step is done synchronously or not, the providing scheduler accepts the agreement and stores it in the agreement store, making it available to both signing parties.
6. The providing scheduler gathers necessary information, namely the machine characteristics for the delegated system in case of resource delegation or the activity state in case of activity delegation. In case of the latter, the step of the protocol end here.
7. For resource delegation, the requesting scheduler is now able to use the delegated resource like its own: For potential jobs that are run on the newly available resource, it ensures the availability of data through regular stage-in mechanisms via the service interface initially negotiated and eventually described in the GLUE document.
8. Jobs are regularly submitted to the resource that is available, using JSDL described templates.
9. Monitoring of the service is done via a monitoring service, exposing the dynamic state of the delegated resource.
10. After job completion, data is staged-out again.

5 Virtualization Approach for the Delegation of Resources

While the management of the federated delegation ends at the protocol level for activity delegation, it is obvious that, to ensure delegated resources to the requesting schedulers, it takes additional steps on the infrastructure level. This does not only comprise the architecture itself but also lower-level details. It is therefore necessary to detail the scenario within which resource delegation takes place and to formulate administrative requirements to achieve this.

5.1 Scenario of the Resource Delegation

In Cloud computing environments, the delegation of resources is a straightforward process: Given the typical IaaS scenario, the customer negotiates with a Cloud provider over a defined number of resources. These are usually accessible through a provider-specific interface and get provisioned as virtual machines on top of a hypervisor. The virtual machines—containing the software execution environment—are either provided by the customer (the prevalent mechanism) or via a software repository offered by the Cloud provider (providing appliances for certain purposes, which can be customized further). Lifecycle management of the machine and deployment of activities or services is completely under the responsibility of the customer.

This approach is fundamentally different from the standard Grid computing scenario: Here, an activity (usually a job) is given to a Grid provider; lifecycle management of this activity is done by the provider. This makes the idea of resource delegation very interesting from a capacity management point of view: It allows a foreign scheduler, for example, servicing a certain user community such as climate research or plasma physics, to incorporate Grid resources from another scheduler's domain to its set of managed resources.

With the DGSI approach, we adopt the ideas and principles from Cloud computing to Grid infrastructures. Overall, we assume the following process: Before confirming to an SLA, see Section 4.3, the providing scheduler (PS) ensures through a Delegation Daemon (DD) running at the delegating site that the resources are provisioned with respect to the Service Description Terms (SDT). The requesting scheduler (RS) is then able to use the delegated resources via a virtualized, SDT-compliant, on-demand Grid middleware (such as Globus GRAM, gLite CE, UNICORE xJS). The SDT-specific middleware

allows the requesting scheduler to use the delegated resources in the same way as it uses its community resources. Through this virtual front-end (VFE), the RS delivers its workload to the resources.

5.2 Administrative Requirements

Both Grid resources and Grid workload are usually not virtualized. The resulting heterogeneity of the infrastructures is one of the most serious challenges of the resource delegation approach. It is therefore necessary to describe additional administrative requirements that an infrastructure such as DGSI has to comply with in order to allow a delegation of resources within a DCI federation.

5.2.1 Trust and Security for Virtual Front-End Systems

Security is an important issue for the virtualized resource delegation approach as well as for every other distributed protocol. A main issue is the responsibility for the security of the virtual machine images. If the requesting community is responsible, then why would the providing Grid community trust them? This means that, if a new virtual Grid middleware is created, this image has to be certified by a trusted entity and stored on an accessible server. In the DGSI approach, this issue is solved by following the software repository approach from Cloud computing: Middleware images are managed and provisioned by the system owner, thus allowing him to keep them under his control.

5.2.2 Firewall

A Grid provider typically has no virtualization software installed on the computing nodes. Therefore, it is likely that the providing site (i.e., the one that supports resource delegation) will run one or more servers that are able to host the VFEs. However, the number of VFEs deployed can vary, and each VFE must (usually because of internal requirements of the various Grid middlewares) have its own IP and needs connectivity with the requesting schedulers. In the following, we will assume that there is a known IP range dedicated to the virtual machines for resource delegation.

Allowing the setup and execution of a virtual machine to be directly accessible from the Internet will not be accepted in most cases. Therefore, it is necessary to have a separate firewall that only allows connections between the VFEs and their corresponding requesting schedulers. Consequently, the firewall should either allow all connections on specific ports on the IP range of the virtual machines from outside or dynamically allow connections from the IP addresses of the requesting schedulers to their assigned VFEs. In both cases, the requesting schedulers should receive information about the address of the VFE that they should connect to.

The latter solution, the more secure one, requires that the IP addresses are known, that the firewall can be updated dynamically, and that the IP address of the requesting scheduler is provided during negotiation.

5.2.3 Certification, Principal, and Naming Administration

Especially with respect to Grid environments, identity management is a major issue as well. A new user and certificate might have to be set up for a machine, and the users and DNs of the requesting community have to be set up by the PS.

Server Certificates Each VFE needs a certificate for the virtualization approach. Usually, this does not provide a challenge if the resource providing scheduler sets up the VM and can therefore use a prepared certificate matching the Domain Name System (DNS) name assigned to the virtual middleware. In case the requesting scheduler provides the VM image, it is the RS' task to arrange the certificate, which is a more dynamic yet elaborate approach.

Principal Management The providing scheduler has to handle the user certification. If the certificate of the VFE is provided by the RS, the VFE might be able to load the user DNs via updates from a VO Membership Registration Service (VOMRS) of the RS. This is challenging as the VFE certificate has to be accepted

by the VOMRS site. However, for many Grid environments, this information is not delivered instantaneous, but rather scheduled updates (e.g., once per day) are run, leaving a significant gap in time until the user is known to the underlying infrastructure. Nevertheless, the resources are to be delegated to the RS at once. Support for short-lived certificates in this scope might provide a solution for this challenge. Interfacing with the VOMRS would be a possible solution as well.

5.2.4 Naming Administration

Each VFE has to receive an IP address when it is started, and this IP has to be known by the RS. Nevertheless, relying on dynamic DNS services is a challenge as reverse DNS lookups can fail. A port forwarding solution using a proxy for this purpose could solve this problem by forwarding connections from the requesting scheduler to the assigned VFE. The scheduler only needs to know the address of the proxy and the port to connect to.

5.3 Software Architecture

From our example use cases, we assume that the resource delegation is negotiated between two meta Community Grid meta schedulers. After the negotiation process, the providing scheduler uses the so-called *Delegation Daemon* (DD), which is running at the delegating site and supervising the delegation and provisioning. When the DD has successfully set up a delegation, it creates a GLUE document that contains a detailed description of the delegated resources. This document is linked to the SLA and is accessible through the WSAG protocol. The RS can extract all necessary information to use the resource from this document.

Overall, the Delegation Daemon poses the core service for realizing the delegation of resources, and its components are described in the following.

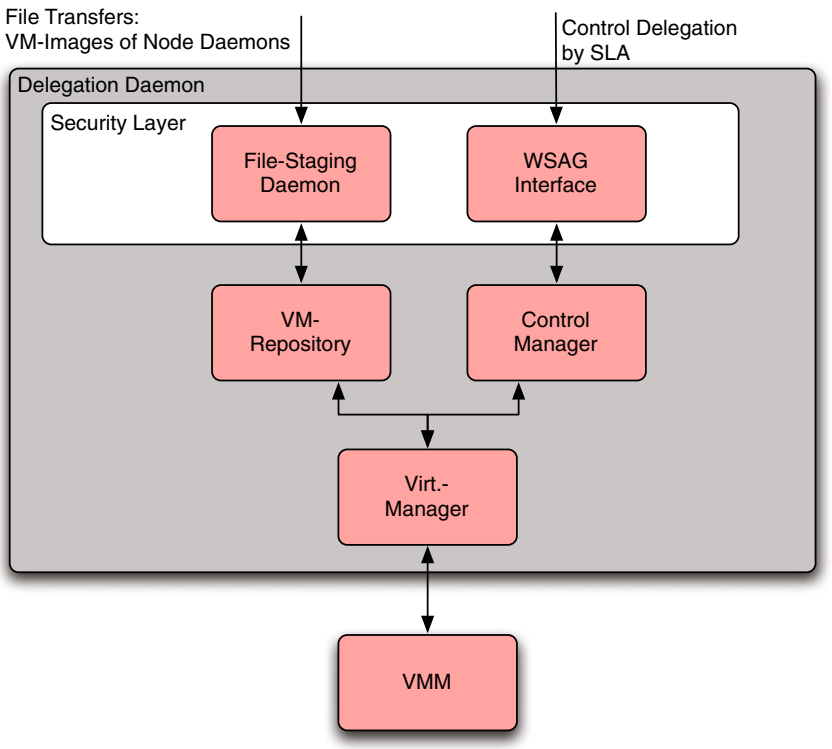


Fig. 5 General architecture of the Delegation Daemon (DD), exposing interfaces for SLA management, file staging, and VM storage, provisioning, and control

For a general overview of the interdependencies of its various components, see Fig. 5.

5.3.1 The WS-Agreement Interface

The WS-Agreement interface exposes means for delivering end-to-end SLA capabilities and allows access for managing specific agreements. In particular, it offers services to the PS in order to handle delegations. More specifically, the following methods are provided:

Management of reservations allows creating, deleting, and modifying reservations on the lower-level infrastructure. Especially for Grid environments, this provides unified means for mapping Service Terms on the abstract SLA level to resource reservations on the concrete LRMS level.

Management of delegations allows creating, deleting and modifying delegations on the higher level infrastructure.

Monitoring of delegated resources exposes monitoring data to the RS or third party services in order to assess the service quality provided through the delegation and monitor the Guarantee Terms for their fulfillment.

Information on delegation and reservation provide a GLUE schema-style document, which describes the delegated resources in detail.

5.3.2 The Control Manager

The control manager is responsible for the management of the whole resource delegation process and provides a persistence layer for information recovery due to unexpected system outages. It provides methods for

Firewall configuration, including management of rules, on-demand hole punching setup, dynamic tunneling for higher-level services, the acceptance of new VOs, and the setup of users and workspaces. The control manager has the necessary information concerning the delegated resource after the setup of the delegation has been successfully completed. The control manager stores the information in a GLUE document and is thus able to access it in the future to extract all information about this delegation. This document will also be transferred to the RS to transmit all necessary usage information.

Identity management taking care of the acceptance and incorporation of new VOs, the creation and mapping of new users, and the setup of workspaces;

LRM interfacing performing reservation setup, handling, and enforcement towards the underlying Local Resource Manager; and

Domain monitoring, including the supervision of the firewall (including opened ports) and provisioned virtual machines.

In addition, the Control Manager takes care of Virtualization Manager steering and the creation of the GLUE schema document for a resource delegation throughout the setup process.

5.3.3 The Virtualization Manager

The task of the virtualization manager is to start, monitor, and stop the virtual front-ends. A possible option to ensure this would be the management through state-of-the-art Cloud management environments, such as Eucalyptus [14], Nimbus [28], or OpenNebula [29]. However, the high complexity of such tools with many additional software dependencies outweighs the advantage of using already developed software, since a lot of functionality is deemed unnecessary for the resource delegation purposes within DGSI. Therefore, it was decided to develop a DGSI-specific virtualization manager that offers the exact functionality needed for resource delegation on the basis of the widely adopted and de-facto standard `libvirt` library [9, 26]. Through this approach, the Virtualization Manager offers functionality for

Image copy ensuring the availability of virtual machine images on the VFE node on-demand;

LRMS adaptation performing reconfiguration of the LRM layer with respect to the requested delegation;

Virtual machine setup taking care of the delegation instance-specific customization of the VFE virtual machine;

Monitoring aggregation delivering aggregated information to higher-level services, regarding the state of the VM system and the LRM.

5.3.4 The Virtual Machine Repository

The VM repository stores the virtual machine images. This enables a requesting scheduler, which frequently uses delegated resources, to reuse an already copied virtual machine image. The VM repository will contain standard Grid middleware installations, including Globus 4.0.x and Globus 4.2.x, UNICORE 5 and UNICORE 6.x, and gLite. In addition, the repository can be used to store additional software such as the resource management and scheduling system, which is to be started on the worker nodes as delegated resource LRM managers.

5.3.5 The File Staging Daemon

The file staging daemon allows the RS to transfer and store the VM images inside the repository. Although being transport-agnostic, we will provide an implementation on the basis of GridFTP for this purpose. A technical issue for the DD is that, due to security requirements, it has to run under a specific user account, which has to be available on all delegated resources and needs administrative privileges to manage the LRMS daemons on the nodes. In modern systems, however, this is not a severe problem: RBAC[2] mechanisms ensure that only certain privileged operations can be issued, and impersonation tools like `sudo` allow the execution of certain commands as another user without compromising administrative principal account details.

Overall, the delegation process can be depicted as a protocol sequence incorporating the different services. In our example (see Fig. 6), the requesting community chooses the Grid middleware in the virtualization approach. The DD at the provider starts the image, provides the IP and ports accessible in the firewall, and reserves the resources.

5.4 Management of the Virtualized Grid Middleware

The virtualization approach offers a high flexi bility as resources can be prepared with an arbitrary middleware. Initially, resource providers supply a set of predefined virtual middleware environments to satisfy common demands. Alternatively, the requesting scheduler can be allowed to set up a custom middleware. If the customer has special demands concerning the delegated resource, he can submit a specialized middleware and use the resources through this interface. This scenario is relevant if the scheduler needs special extensions to the middleware or a certain LRMS including workflow orchestration integration. The virtualization approach does not imply code changes in the standard Grid middleware environments. Therefore, if a scheduler does not support the interface provided by the proxy approach, it can still use the virtualization approach for participating in resource delegation.

5.4.1 Realization

The key idea is that the (existing) front-end is responsible for the start of the virtual front-end (VFE), including the virtual Grid middleware. For this, a delegation daemon with a WSAG interface extends the front-end. After the negotiation between the Grid schedulers, the providing Grid scheduler hands over the resulting SLA to the Delegation Daemon on the front-end node. Therefore, the requesting scheduler has to be able to communicate with the daemon's WSAG interface. The Delegation Daemon starts the VFE prior to the beginning of the negotiated time slice. The job submission and monitoring are then done by the middleware interface of the virtual front-end. After the end of the negotiated time slice, the Delegation Daemon shuts down the VFE.

[2]Role-Based Access Control

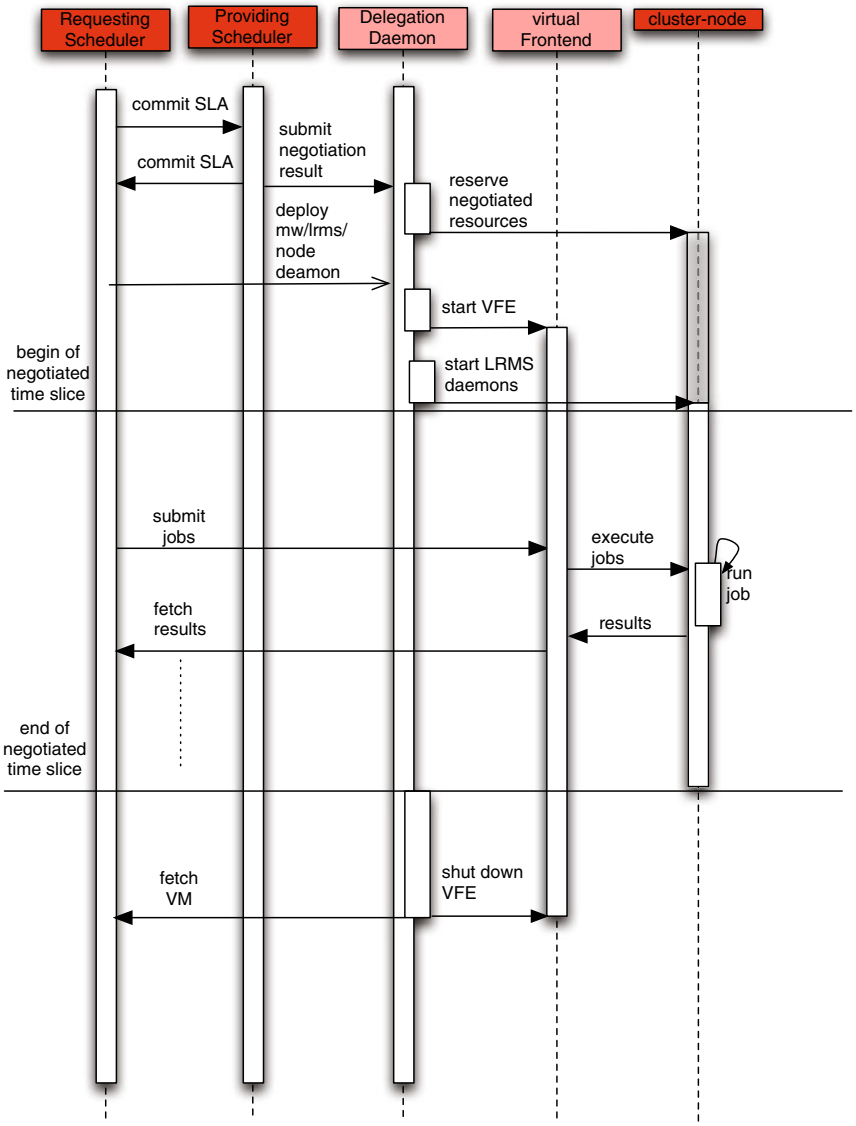


Fig. 6 The sequence of a resource delegation on the basis of the aforedescribed virtualization approach. After negotiation, the DD (using the components as described above) has to deploy the middleware infrastructure and start the VMs. At the start time of the negotiated delegation, the LRM daemons are started on the underlying compute nodes. The requesting scheduler can then access the resources through the newly deployed services. At delegation end time, all services are automatically shut down

5.4.2 Basic Setup of the Middleware Image

The resource provider has two options for provisioning the VFE: It can either be started as a VM on an existing front-end or on one of the delegated compute nodes. The first method allows distributing real IP addresses, so the virtual front-end can be contacted from any Internet client and, therefore, from the scheduler of the requesting community. The second option, which is more scalable, is to not use a dedicated server but one of the delegated worker nodes as virtual front-end. On most sites this implies to have an internal IP in a network address translation (NAT) environment. In this case the virtual front-end has to set up a tunnel into the network of the requesting community, and the tunnel has to use a real IP address out of the range of the requesting community's network. IP administration and certificate handling have to be handled by the requesting community. Virtually, the front-end runs in the environment of the requesting community.

The existing front-end running the Grid middleware is often not identical with the host that runs the LRMS. For instance, in the D-Grid Reference Installation [1], there is no connectivity between the front-end and the compute nodes. However, this is presumably necessary for the VFE depending on the setup of the LRMS used within a delegation.

The VM image of the virtual front-end could either be a predefined image provided by the resource provider or a custom image provided by the requesting community. In the first case, the site provides a set of VMs running standard Grid middleware environments. The second case provides much more flexibility for the requesting community but constitutes a security risk for the providing site because the VFE has access to the internal network. Additionally, this case always implies that the requesting community gets administrative privileges on the VFE. Therefore, the decision to support this scenario has to be carefully weighted by each site.

The resource provider or the requesting community can provide the DNS name and the corresponding host certificate of the VFE. The latter takes place in case the requesting community also provides the VM image of the VFE. The providing party offers the IP address of the VFE and should reserve a set of IP addresses to support multiple resource delegations.

It is necessary that the certificates involved in a resource delegation process, e.g., of the requesting scheduler, the VFE, and users are issued by a CA the resource provider trusts. For the first generation implementation inside the D-Grid infrastructure, it is also required that the requesting VO and its users already exist at the providing site. There are several options to handle the management of the LRMS within a delegation scenario. One option is to start a new LRMS server (e.g., a PBS server) on the VFE, which helps to separate the delegated resources from the non-delegated ones. The LRMS daemons (e.g., the `mom` in PBS-based systems) running on the worker nodes are either the daemons running on the delegated nodes or newly started job daemons. In the first case, the existing daemons have to be disconnected from their server, which not only is very intrusive from the siteÅŘs perspective but might also be complicated. In the second case, the new LRMS daemons could run parallel to the existing daemons or they could be started as jobs by the existing daemons. The latter is the preferred approach because using the existing daemons to start the new infrastructure ensures that the new infrastructure is automatically destroyed when the negotiated time slice ends.

Another aspect is the data management on the delegated resources. In accordance with the D-Grid Reference Installation, the VFE must provide home directories for the users of the requesting community, which must also exist on the compute nodes. Additionally, scratch directories are necessary. For the first generation implementation, the existing directories are used just as on the normal front-end. If these directories are mounted on the compute nodes, the job results are automatically available on the VFE. Furthermore, the virtualization approach can also provide resources with additional application software. To this end, the VFE could export directories to the compute nodes within a distributed file system. Alternatively, VMs running on the compute nodes could already contain necessary software. At the end of the negotiated time slice, the Delegation Daemon can wait a short period of time before it shuts down the VFE. The requesting scheduler can use this time for final data transfers. If a custom VFE was used, the requesting scheduler might also want to retrieve the VM image after shutdown.

Eventually, a delegation setup is delivered as depicted in Fig. 7.

5.5 Employed Technologies

As Section 5 discusses several alternative approaches that are generally suitable to implement the resource delegation process, the following paragraph gives a brief overview of the technologies chosen for the implementation in DGSI.

For the implementation and steering of the virtual machines, we use `libvirt`, which allows the transparent management of several virtualization technologies, like Xen, VMware ESX, KVM, and VirtualBox. Within the project, we focus on *Xen* and *KVM*. The WS-Agreement implementation is based on the *WSAG4J* Java framework. The

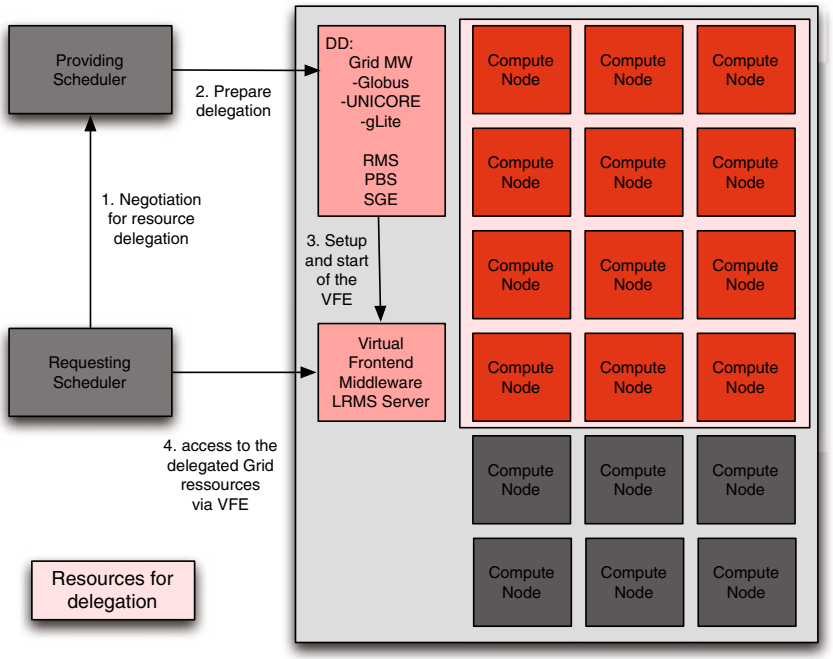


Fig. 7 A delegation scenario based on a virtualized front-end node. After negotiating the conditions of the delegation (step 1), the providing scheduler notifies the Delegation Daemon to prepare the delegation (step 2), which in turn sets up and starts the virtual front-end. Then, the requesting scheduler can access the delegated resources through the middleware as requested in the Service Description Terms

connection of these through the Virtualization Manager is delivered by the `libvirt` *Java API bindings*.

In DGSI, we focus on *Globus Toolkit 4* and *UNICORE 6* as supported middleware systems. On the LRMS level, we (although being technology-agnostic) support *PBS (TORQUE/ Maui)*, /emphLSF, and *SGE*.

6 Information Model

The DGSI information model is designed to support the three phases of a delegation: discovery of resources and meta-schedulers, negotiation, and monitoring. This includes a description of resources, meta-schedulers, constraints, monitoring data and configuration state of resource delegations. For example, the constraints of a delegation request may require four machines with 8 CPU cores, 2GB memory/CPU core and 10GB temporary disk space.

In DGSI, we use GLUE 2.0 [2] as a basis for resource descriptions. GLUE annotates resources with both static configuration (e.g., machine properties such as total amount of memory and number of cores) as well as dynamically changing attributes including system queue length or current memory usage. The description of resources is necessary for three parts of the delegation protocol:

1. Meta-scheduler discovery,
2. matching the capabilities of a resource with the requested resources, and
3. static configuration.

First, resource descriptions are aggregated to describe the capabilities of a single meta-scheduler and used for meta-scheduler discovery. A simple strategy for discovering an appropriate scheduler is to use a global resource registry for all meta-schedulers. This approach, however, introduces a single-point of failure and is not scalable. We are currently investigating how resource discovery can be done with an eventually consistent, low traffic, gossiping protocol [23].

Second, as part of the WS-Agreement protocol (see Fig. 4), each providing scheduler maintains a set of templates indicating its capabilities. In DGSI, these templates are in JSDL [4] extended

with GLUE to provide the user with a more flexible resource description format. Finally, we use GLUE to describe the configuration of the delegated resource. This information is passed to the requesting scheduler via the WS-Agreement protocol as part of a template.

JSDL is used by the requesting scheduler to model the requirements of the resource. However, since JSDL only provides a limited number of attributes for describing resources, we extend JSDL with a GLUE document.

To complete our information model, we need a way to describe time constraints of delegations. Since this is not part of JSDL or GLUE, we use the Advance Reservation Profile (ARP) currently developed by the GRAAP-WG in the OGF.[3] The ARP allows simple reservations of the type "3 hours anywhere between 15:00 to 23:00".

7 Performance Evaluation

The prototype for the resource delegation approach described in this paper has been implemented in generation one of the DGSI virtualization approach. The prototype demonstrates the general feasibility of resource delegations. The managing software for the resource delegation is based on a Java implementation of WS-Agreement (WSAG4J) [40]. The prototype can reserve the requested resources on a torque/maui RMS and create and manage KVM based virtual front-ends. Abilities of the control manager, like dynamic firewall configuration and identity management, have not yet been implemented. The necessary firewall-ports have to be configured statically, and the necessary certificates have to be deployed manually. The file-staging daemon is not available, and the VM-Repository is limited. The first prototype VFE supports Globus Toolkit 4.0.8 images.

This section presents performance measures and evaluates the abilities and throughput of a resource delegation. The test scenario had three load tests.

1. load test with 100 times one RD negotiation
2. load test with 100 times two parallel RD negotiations
3. load test with 100 times four parallel RD negotiations

Every negotiation was encapsulated in a java *run()* method. The time for the negotiation was measured taking the system time before and directly after the negotiation request. The negotiation was requested from the Paderborn Center for Parallel Computing. The goegrid cluster system in Göttingen created the reservation. The results of the negotiation tests are shown in Fig. 8.

This border marks in the figure are no confidence intervals. Through a lot of measurements, confidence intervals become very narrow, even when the measurements are scattered, because for many measurements there is a high probability that the average is beneath the calculated average. We want to show that the measurements are varying; the borders show the area where 95% of the measurements fall into. The results show that the duration of a negotiation

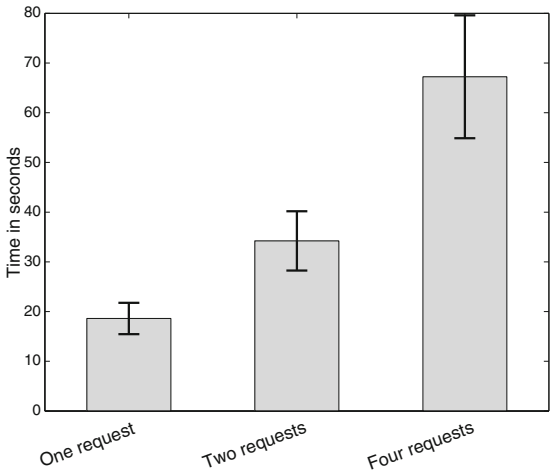


Fig. 8 The duration of the negotiations for a resource delegation

[3]The Advance Reservation Profile Document is under active development on the group's collaboration space, but yet to be published as a full proposed recommendation document.

increases linearly in the number of parallel executions. The average duration of the tests was:

1. battery: 18.609 ms
2. battery: 34.221 ms
3. battery: 67.235 ms

The long duration is caused by the time the VFE reservation and setup and the maui reservation take. The VFE creation includes the cloning of a virtual machine that uses several seconds. The VFE creation and the reservations have problems with parallel execution and are synchronized, explaining the increasing execution time during the parallel negotiations. To underline this, Fig. 9 shows server internal measurements of the negotiation process.

– Strategy duration: mean of 20.271 ms
– RMS reservation: mean of 44.556 ms
– VFE reservation: mean of 46.848 ms
– Handler duration: mean of 30.949 ms
– VFE setup: mean of 12.653 ms

The server side negotiation process uses two methods. Firstly the *strategy* method, which reserves the resources on the RMS and a time slot for the VFE. Secondly, the *handler* method is called and starts the VFE setup when the strategy method successfully reserved the resources. Thus, in Fig. 9 the *Strategy duration* includes the duration time for creating the *RMS reservation*. The *Handler duration* includes the time for running the *VFE setup*. Again, the figures indicating the area 95% of the measurements fell into. One can see that the *VFE setup* of a RD creation takes 12 seconds on average. The *handler duration* takes between from 12 to 50 s. This is due to the fact that the call of the *VFE setup* is synchronized and leads to a huge delay of the handler duration when parallel RD negotiations are called. The effect for the *Strategy duration* is similar. The huge percentiles show that single negotiations are performant and that, due to serialization, the duration increases by concurrent negotiations'. This underlines why the single RD negations are much faster than concurrent negotiations executions.

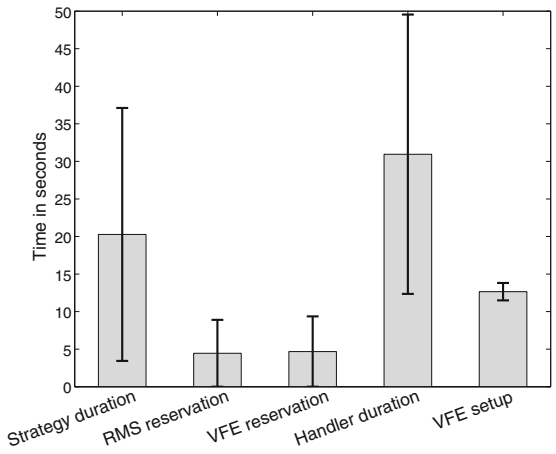


Fig. 9 The duration of the server side components

However, the first prototype demonstrates that RD negotiations are possible. For the next generation of the software, we plan to migrate the negotiation process to the software stack, of the SLA4DGrid project, which includes a sophisticated WS-Agreement stack, and to submit the *VFE setup* as a separate job. This process should improve the responsiveness of the negotiation process because the duration of the negotiation in the generation two RD software stack will consist of the strategy method only.

8 Conclusion and Future Work

Driven by user communities and resource providers, DGSI implements two use-case models: Delegation of activities and delegation of resources; both were not available before in D-Grid. Whereas delegation of activities means job-forwarding at this point in time, delegation of resources implements a Cloud usage style for Grid resources. The control on a set of resources is handed over to the user for the negotiated time. The genuine innovation of DGSI is the definition of the protocol suite to negotiate delegations on activities (jobs) and resources. To describe these protocols, DGSI harvests on Grid standards and delivers a confirmation of value of systematic OGF work over many years. The DGSI functionalities are achieved by reuse of proven technologies in a practical, straightforward way. Basically, DGSI recombines a number of

reference implementations and meta-schedulers. Besides implementing missing parts of the architecture and providing adequate security mechanisms for resource delegation, at the heart of the practical technical work for the creation of creating some GLUE(ing) elements, JSDL profiles and WS-Agreement templates, and the performance of scrutinized QA testing to achieve production readiness. In order to show real-world applicability, we have evaluated the described approach in a real-world testbed, and recorded promising performance results.

DGSI is Path-breaking for Grid and Cloud Future
Originally, DGSI was set up as a gap project for D-Grid, closing identified functionality gaps within the multi-technology environment of the German e-science Grid. DGSI achieves this goal by strict (re-)use of standards, enabling all technology providers (be it Grid or Cloud middleware) to adopt the methods and protocols and implement negotiated delegations. By experience, the Grid community can state: There will never be a big enough Grid. The same applies to Clouds, where other criteria add to the picture. DGSI realizes the integration of Grid and Cloud technology by providing the technology to dynamically extend a local Grid with remote resources hosted in a different administrative domain. For both, Grids and Clouds, nobody seriously envisions a future where most or even all run the same software stack, using the same middleware. Technological diversity will dominate—which is good: systems biology taught us about the stabilizing effects of diversity on ecosystems. DGSI delivers the blue-print to utilize technological diversity. The standards-based methods on negotiation and delegation do not limit the creativity of the Cloud or Grid middleware developers while providing interoperability with other technology stacks (e.g. legacy).

DGSI's current agenda reaches to the point where several meta-schedulers and LRMS are delegation-enabled and those methods are introduced into D-Grid operations. Through continuous active contributions to the OGF working groups, DGSI takes care to keep innovative elements in line with standardization. Beyond or beside the current project and as part of future work, we are looking for partners to work on Cloud stacks and potentially more Grid technologies. DGSI blue-prints may be the starting point and base line for interoperability in diversity. The DGSI partners are offering help to implement the related protocols and standards and to enhance delegation methods to suit innovative use cases.

We envision a world of distributed computing with resources from many different providers using various technologies, where borders between Clouds and Grids are blurred or vanishing, where new technologies are introduced without service interruption, where innovation is free to flourish while staying productively connected, fully interoperable.

References

1. Alef, M., Fieseler, T., Freitag, S., Garcia, A., Grimm, C., Gürich, W., Mehammed, H., Schley, L., Schneider, O., Volpato, G.: Integration of multiple middlewares on a single computing resource. Future Gener. Comput. Syst. **25**(3), 268–274 (2009). doi:10.1016/j.future.2008.05.004
2. Andreozzi, S., Burke, S., Ehm, F., Field, L., Galang, G., Konya, B., Litmaath, M., Millar, P., Navarro, J.: GLUE Specification v. 2.0 (2009). Grid Forum proposed recommendation GFD.147, available at http://www.ogf.org/documents/GFD.147.pdf
3. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Grid Forum proposed recommendation GFD.107, available at http://www.ogf.org/documents/GFD.107.pdf
4. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: Job Submission Description Language (JSDL) Specification v1.0 (2008). Grid Forum recommendation GFD.136, available at http://www.ogf.org/documents/GFD.136.pdf
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. rep., EECS Department, University of California, Berkeley (2009)
6. Battré, D., Hovestadt, M., Wäldrich, O.: Lessons learned from implementing WS-Agreement. In: Wieder, P., Yahyapour, R., Ziegler, W. (eds.) Grids and Service-Oriented Architectures for Service Level Agreements, pp. 23–34. Springer US (2010). doi:10.1007/978-1-4419-7320-7_3
7. Baur, T., Breu, R., Kálmán, T., Lindinger, T., Milbert, A., Poghosyan, G., Reiser, H., Romberg, M.: An interoperable grid information system for

integrated resource monitoring based on virtual organizations. J. Grid Computing **7**, 319–333 (2009). doi:10.1007/s10723-009-9134-3
8. Bazaar: Project Website: https://www.ce-egee.org/weblog/bazaar (2010)
9. Bolte, M., Sievers, M., Birkenheuer, G., Niehoerster, O., Brinkmann, A.: Non-intrusive virtualization management using libvirt. In: Proceedings of Design, Automation and Test in Europe (DATE). Dresden, Germany (2010)
10. BREIN—Business Objective Driven Reliable and Intelligent Grids for Real Business: Project Website: http://www.eu-brein.com (2010)
11. Cornwall, L.A., Jensen, J., Kelsey, D.P., Frohner, Á., Kouřil, D., Bonnassieux, F., Nicoud, S., Lőrentey, K., Hahkala, J., Silander, M., Cecchini, R., Ciaschini, V., dell'Agnello, L., Spataro, F., O'Callaghan, D., Mulmo, O., Volpato, G.L., Groep, D., Steenbakkers, M., McNab, A.: Authentication and authorization mechanisms for multi-domain grid environments. J. Grid Computing **2**, 301–311 (2004). doi:10.1007/s10723-004-8182-y
12. England, D., Weissman, J.B.: Costs and benefits of load sharing in the computational Grid. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) Proceedings of the 10th Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science (LNCS), vol. 3277, pp. 160–175. Springer (2004)
13. Ernemann, C., Hamscher, V., Schwiegelshohn, U., Streit, A., Yahyapour, R.: On advantages of Grid computing for parallel job scheduling. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID02), pp. 39–46. IEEE Press (2002)
14. Eucalyptus Systems: Eucalyptus. Website: http://open.eucalyptus.com/ (2010)
15. Field, L., Laure, E., Schulz, M.: Grid deployment experiences: Grid interoperation. J. Grid Computing **7**, 287–296 (2009). doi:10.1007/s10723-009-9128-1
16. Fölling, A., Grimme, C., Lepping, J., Papaspyrou, A.: Decentralized Grid scheduling with evolutionary fuzzy systems. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) Proceedings of the 14th Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science (LNCS), vol. 5798, pp. 16–36. Springer (2009)
17. Fölling, A., Grimme, C., Lepping, J., Papaspyrou, A., Schwiegelshohn, U.: Competitive co-evolutionary learning of fuzzy systems for job exchange in computational Grids. Evol. Comput. **17**(4), 545–560 (2009)
18. Gagliardi, F., Jones, B., Grey, F., Begin, M.E., Heikkurinen, M.: Building an infrastructure for scientific Grid computing: status and goals of the EGEE project. Phil. Trans., Ser. A, Math. Phys. Eng. Sci. **363**(1833), 1729–1742 (2005)
19. GRAAP-WG—Grid Resource Allocation Agreement Protocol Working Group): Project Website: https://forge.gridforum.org/projects/graap-wg (2010)
20. Grimme, C., Lepping, J., Papaspyrou, A.: Prospects of collaboration between compute providers by means of job interchange. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) Proceedings of Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science (LNCS), vol. 4942, pp. 132–151. Springer (2007)
21. GSA-RG—Grid Scheduling Architecture Research Group: Project Website: https://forge.gridforum.org/projects/gsa-rg (2009)
22. Iosup, A., Epema, D.H., Tannenbaum, T., Farrellee, M., Livny, M.: Inter-operating Grids through delegated matchmaking. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC07). Reno, NV (2007)
23. Iyengar, V., Tilak, S., Lewis, M.J., Abu-Ghazaleh, N.B.: Non-uniform information dissemination for dynamic Grid resource discovery. In: NCA '04: Proceedings of the Network Computing and Applications, Third IEEE International Symposium, pp. 97–106. IEEE Computer Society, Washington, DC (2004)
24. Kota, S.R., Shekhar, C., Kokkula, A., Toshniwal, D., Kartikeyan, M.V., Joshi, R.C.: Parameterized module scheduling algorithm for reconfigurable computing systems. In: ADCOM '07: Proceedings of the 15th International Conference on Advanced Computing and Communications, pp. 473–478. IEEE Computer Society, Washington, DC (2007)
25. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Scheduling jobs on the Grid—multicriteria approach. Comput Methods Sci Technol **12**(2), 123–138 (2006)
26. Libvirt Development Team: Libvirt. Project Website: http://libvirt.org/ (2010)
27. NextGRID SLA Schema Generalized Specification: http://www.nextgrid.org/GS/management_systems/SLA_management/NextGRID_SLA_schema.pdf (2009)
28. Nimbus Community: Nimbus. Project Website: http://www.nimbusproject.org/ (2010)
29. OpenNebula Community: OpenNebula:. Project Website: http://www.opennebula.org/ (2010)
30. Parkin, M., Badia, R.M., Martrat, J.: A Comparison of SLA Use in Six of the European Commissions FP6 Projects. Tech. Rep. TR-0129, Institute on Resource Management and Scheduling, CoreGRID—Network of Excellence. URL http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0129.pdf (2008)
31. PHOSPHORUS—Lambda User Controlled Infrastructure For European Research:. Project Website: www.ist-phosphorus.eu (2010)
32. Riedel, M., Terstyanszky, G.: Grid interoperability for e-research. J. Grid Computing **7**, 285–286 (2009). doi:10.1007/s10723-009-9138-z
33. Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schulz, S., Stokes-Rees, I.: Grid and cloud computing: opportunities for integration with the next generation network. J. Grid Computing **7**, 375–393 (2009). doi:10.1007/s10723-009-9132-5
34. Rumpl, A., Wäldrich, O., Ziegler, W.: Extending WS-Agreement with multi-round negotiation capability. In: Grids and Service-Oriented Architectures for Service Level Agreements, CoreGRID series 13, pp. 89–103. Springer (2010). doi:10.1007/978-1-4419-7320-7_9
35. Seidel, J., Wäldrich, O., Wieder, P., Yahyapour, R., Ziegler, W.: SLA for resource management and scheduling—a survey. In: Grid Middleware and Ser-

vices: Challenges and Solutions, CoreGRID Series 8. Springer (2008)
36. SLA@SOI—Project Website:. Project Website: http://sla-at-soi.eu (2010)
37. SmartLM—Grid-friendly Software Licensing for Location Independent Application Execution:. Project Website: http://www.smartlm.eu (2010)
38. SORMA—Self-Organizing ICT Resource Management:. Project Website: http://www.im.uni-karlsruhe.de/sorma (2010)
39. Subramaniyan, R., Troxel, I., George, A.D., Smith, M.: Simulative analysis of dynamic scheduling heuristics for reconfigurable computing of parallel applications. In: Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays, pp. 230–230. ACM (2006)
40. Wäldrich, O.: WS-Agreement for JAVA (WSAG4J): Project Website: http://packcs-e0.scai.fraunhofer.de/wsag4j/
41. IBM Web Service Level Agreements (WSLA) Project: Project Website: http://www.research.ibm.com/wsla
42. OASIS Web Services Resource Framework (WSRF) TC: Project Website: http://www.oasis-open.org/committees/wsrf

A. Annex

A.1 Acronyms

The numbers refer to the page where the acronym was first used.

ACL Agent Communication Language . . . 34
API Application Programming Interface . . . 7
ARGON Allocation and Reservation of Grid-enabled Optical Networks . . 25
ASP Application Service Provider . . . 10
BCR Binding Corporate Rule . . . 5
BMBF Federal Ministry of Education and Research . . . iv
BMWi Federal Ministry for Economic Affairs and Energy . . . 196
BSLA Binding Service Level Agreements . . . 38
CISG Convention on Contracts for the International Sale of Goods . . . 39
CNP Contract Net Protocol . . . 34
CONNECT Communications Networks, Content and Technology . . . 31
CP Control Plane . . . 45
CPU Central Processing Unit . . . 22
CRDL Computing Resource Definition Language . . . 24
CSCS Swiss National Supercomputing Centre . . . 46
C-SIG SLA Cloud Select Industry Group – Subgroup on Service Level Agreements . . . 27
DD Delegation Daemon . . . 57
DFN Deutsches Forschungsnetz . . . 44
DG Directorate General . . . 31
DGSI D-Grid Scheduler Interoperability . . . 5
DMTF Distributed Management Task Force . . . 29
DPD Data Protection Directive . . . 54
DSL Domain-Specific Language . . . 21
EC European Commission . . . iv
EC2 Elastic Compute Cloud . . . 10
EEA European Economic Area . . . 54
EMOF Essential Meta Object Facility . . . 20
EPFL École polytechnique fédérale de Lausanne . . . 46
FIPA Foundation for Intelligent Physical Agents . . . 34
GGJM Global Grid Job Manager . . . 24
GMD Gesellschaft für Mathematik und Datenverarbeitung . . . 196
GRAAP-WG Grid Resource Allocation Agreement Protocol Working Group . . 5
GUI Graphical User Interface . . . 47
HES-SO University of Applied Sciences and Arts of Western Switzerland 46
HLRZ Höchstleistungsrechenzentrum . . . 196
HPC High Performance Computing . . . 2
HTC High Throughput Computing . . . 2
IaaS Infrastructure as a Service . . . 5
IANOS Intelligent Application-oriented Scheduling . . . 21
ICNP Iterated Contract Net Protocol . . . 34
IEC International Electrotechnical Commission . . . 11
IEEE The Institute of Electrical and Electronics Engineers . . . 196
IP Infrastructure Provider . . . 28
IPR Intellectual Property Right . . . 5
ISO International Organization for Standardization . . . 11

ISS Intelligent Scheduling System ... 22
ISV Independent Software Vendor ... 55
IT Information Technology ... 2
ITIL IT Infrastructure Library ... 15
JSDL Job Submission Description Language ... 22
JTC 1 Joint Technical Committee for Information Technology ... 31
KPI Key Performance Indicator ... 12
LRMS Local Resource Management System ... 25
MNM Meta-Negotiation Middleware ... 37
MSS Meta Scheduling Service ... 22
MTBF Mean Time Between Failures ... 21
NIST National Institute of Standards and Technology ... 11
NREN National Research Network ... 44
NSP Network Service Plane ... 45
OASIS Organization for the Advancement of Structured Information Standards ... 27
OCCI Open Cloud Computing Interface ... 20
OGF Open Grid Forum ... 5
OGSA Open Grid Services Architecture ... 38
OMG Object Management Group ... 20
OVF Open Virtualization Format ... 29
PaaS Platform as a Service ... 10
QoP Quality of Protection ... 49
QoS Quality of Service ... i
REST Representational State Transfer ... 20
RMS Resource Management System ... 45
RSLA Resource Service Level Agreements ... 38
SaaS Software as a Service ... 10
SBNP Simple Bilateral Negotiation Protocol ... 35
SC Subcommittee ... 31
SCAI Fraunhofer-Institut für Algorithmen und Wissenschaftliches Rechnen ... iv
SCC Standard Contractual Clause ... 5
SDO Standards Developing Organisation ... 11
SDT Service Description Term ... 5
SLA Service Level Agreement ... i
SLM Service Level Management ... 195
SLO Service Level Objective ... 11
SME Small and Medium Enterprises ... 3
SNAP Service Negotiation and Acquisition Protocol ... 38
SP Service Provider ... 28
TMF TeleManagement Forum ... 10
TOSCA Topology and Orchestration Specification for Cloud Applications ... 27
TREC Trust, Risk, Eco-efficiency, Cost ... 30
TSI Target System Interface ... 45
TSLA Task Service Level Agreements ... 38
TTP Trusted Third Party ... 12
TU Technische Universität ... 46

UN	United Nations	39
UPL	UNICORE Protocol Layer	45
USDL	Unified Service Description Language	30
VIOLA	Vertically Integrated Optical Testbed for Large Applications	22
VM	Virtual Machine	49
WAN	Wide Area Network	17
WSAG4J	WS-Agreement for Java	40
WSDL	Web Services Description Language	18
WSLA	Web Service Level Agreement	17
WSRF	Web Services Resource Framework	38
XML	Extensible Markup Language	6

A.2 Service Manifest Schema

Listing A.1: Complete Schema of the Service Manifest

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:opt="http://schemas.optimis.eu/optimis/"
           xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
           targetNamespace="http://schemas.optimis.eu/optimis/"
           elementFormDefault="qualified" attributeFormDefault="qualified">
    <xs:import namespace="http://schemas.dmtf.org/ovf/envelope/1"
               schemaLocation="http://schemas.dmtf.org/ovf/envelope/1/dsp8023_1.1.0.xsd"/>
    <xs:element name="ServiceManifest" type="opt:ManifestType">
        <!--
          componentId MUST BE unique within one service manifest
          for example: VirtualMachineComponent opt:componentId="jboss"
        -->
        <xs:key name="vmComponentKey">
            <xs:selector xpath=".//opt:VirtualMachineComponent"/>
            <xs:field xpath="@opt:componentId"/>
        </xs:key>
        <!-- make sure that each ComponentId element in the Scope section
             REFERENCES to an existing componentId -->
        <xs:keyref name="vmComponentKeyRef" refer="opt:vmComponentKey">
            <xs:selector xpath=".//opt:Scope/opt:ComponentId"/>
            <xs:field xpath="."/>
        </xs:keyref>
    </xs:element>

    <xs:complexType name="ManifestType">
        <xs:annotation>
            <xs:documentation>
                Type definition of the OPTIMIS service manifest.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="opt:ServiceDescriptionSection"/>
            <xs:element ref="opt:TRECSection" minOccurs="0"/>
            <xs:element ref="opt:ElasticitySection" minOccurs="0"/>
            <xs:element ref="opt:DataProtectionSection" minOccurs="0"/>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="strict"
                 namespace="##other"/>
        </xs:sequence>
        <xs:attribute name="manifestId" use="required">
            <xs:annotation>
                <xs:documentation>
                    The manifest id is composed of the SLA name and SLA version. The
                    values are separated by a colon.
                </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern value="\w[\w_\-]*\:\d+"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="serviceProviderId" type="xs:string"/>
    </xs:complexType>

    <xs:element name="ServiceDescriptionSection" type="
         opt:ServiceDescriptionSectionType"/>
    <xs:element name="ServiceComponent" type="opt:ServiceComponentType"/>
    <xs:element name="VirtualMachineDescription" type="
         opt:VirtualMachineDescriptionType"
                substitutionGroup="opt:ServiceDescriptionSection"/>
    <xs:element name="VirtualMachineComponent" type="
         opt:VirtualMachineComponentType"
                substitutionGroup="opt:ServiceComponent"/>

    <xs:element name="TRECSection" type="opt:TRECSectionType"/>
    <xs:element name="ElasticitySection" type="opt:ElasticitySectionTypeY1"/>
    <xs:element name="DataProtectionSection" type="opt:DataProtectionSectionType"/>
```

```
64
65      <xs:complexType name="ServiceDescriptionSectionType" abstract="true">
66          <xs:annotation>
67              <xs:documentation>
68                  Base type of an OPTIMIS Service Description. All service
                        descriptions
69                  inherit from this type. Additional service descriptions MAY be
                        defined
70                  for OPTIMIS and can be included into the service manifest as XSD
71                  substitution group.
72              </xs:documentation>
73          </xs:annotation>
74          <xs:sequence>
75              <xs:element ref="opt:ServiceComponent" minOccurs="1" maxOccurs="
                    unbounded"/>
76          </xs:sequence>
77          <xs:attribute name="serviceId" type="xs:string" use="required"/>
78          <xs:attribute name="isFederationAllowed" type="xs:boolean" use="required"/>
79      </xs:complexType>
80
81      <xs:complexType name="ServiceComponentType" abstract="true">
82          <xs:annotation>
83              <xs:documentation>
84                  Base type of an OPTIMIS Service Component. All service components
85                  inherit from this type. Additional service components MAY be
                        defined for
86                  OPTIMIS and can be included into the service manifest as XSD
87                  substitution group.
88              </xs:documentation>
89          </xs:annotation>
90          <xs:attribute name="componentId" type="xs:string" use="required"/>
91      </xs:complexType>
92
93      <xs:complexType name="AbstractVirtualMachineDescriptionType" abstract="true">
94          <xs:annotation>
95              <xs:documentation>
96                  Provisioning of plain virtual machines is the default OPTIMIS use
                        case.
97                  The VirtualMachineServiceDescription specifies the VMs that are
                        provided
98                  to a customer once an SLA is created.
99              </xs:documentation>
100         </xs:annotation>
101         <xs:complexContent>
102             <xs:extension base="opt:ServiceDescriptionSectionType">
103                 <xs:sequence>
104                     <!--<xs:element ref="opt:VirtualMachineComponent" minOccurs="1"
                            -->
105                     <!--maxOccurs="unbounded"/>-->
106                     <xs:element name="AffinitySection" type="
                            opt:AffinitySectionType" maxOccurs="1"/>
107                     <xs:element name="AntiAffinitySection" type="
                            opt:AntiAffinitySectionType" maxOccurs="1"/>
108                     <xs:any namespace="##other" processContents="strict" minOccurs=
                            "0" maxOccurs="unbounded"/>
109                 </xs:sequence>
110             </xs:extension>
111         </xs:complexContent>
112     </xs:complexType>
113
114     <xs:complexType name="VirtualMachineDescriptionType">
115         <xs:annotation>
116             <xs:documentation>
117                 Provisioning of plain virtual machines is the default OPTIMIS use
                        case.
118                 The VirtualMachineServiceDescription specifies the VMs that are
                        provided
119                 to a customer once an SLA is created.
120             </xs:documentation>
121         </xs:annotation>
122         <xs:complexContent>
123             <xs:restriction base="opt:AbstractVirtualMachineDescriptionType">
124                 <xs:sequence>
```

```
125                     <xs:element ref="opt:VirtualMachineComponent" minOccurs="1"
                            maxOccurs="unbounded"/>
126                     <xs:element name="AffinitySection" type="
                            opt:AffinitySectionType" maxOccurs="1"/>
127                     <xs:element name="AntiAffinitySection" type="
                            opt:AntiAffinitySectionType" maxOccurs="1"/>
128                     <xs:any namespace="##other" processContents="strict" minOccurs=
                            "0" maxOccurs="unbounded"/>
129                 </xs:sequence>
130             </xs:restriction>
131         </xs:complexContent>
132     </xs:complexType>
133
134     <xs:complexType name="ScopedSectionType" abstract="true">
135         <xs:sequence>
136             <xs:element name="Scope" type="opt:ScopeArrayType"/>
137         </xs:sequence>
138     </xs:complexType>
139
140     <xs:element name="OVFDefinition" type="ovf:EnvelopeType"/>
141     <xs:element name="AllocationConstraints" type="opt:AllocationConstraintType"/>
142     <xs:element name="AffinityConstraints" type="opt:AffinityConstraintType"/>
143     <xs:element name="AntiAffinityConstraints" type="opt:AntiAffinityConstraintType
            "/>
144     <xs:element name="ServiceEndpoints" type="opt:ServiceEndpointsType"/>
145
146     <xs:complexType name="VirtualMachineComponentType">
147         <xs:annotation>
148             <xs:documentation>
149                 It is used to describe one particular class of virtual machines
                        that are
150                 deployed in an OPTIMIS IP infrastructure.
151             </xs:documentation>
152         </xs:annotation>
153         <xs:complexContent>
154             <xs:extension base="opt:ServiceComponentType">
155                 <xs:sequence>
156                     <xs:element ref="opt:OVFDefinition"/>
157                     <xs:element ref="opt:AllocationConstraints"/>
158                     <xs:element ref="opt:AffinityConstraints"/>
159                     <xs:element ref="opt:AntiAffinityConstraints"/>
160                     <xs:element ref="opt:ServiceEndpoints" minOccurs="0"/>
161                 </xs:sequence>
162             </xs:extension>
163         </xs:complexContent>
164     </xs:complexType>
165
166     <xs:complexType name="AllocationConstraintType">
167         <xs:annotation>
168             <xs:documentation>
169                 Defines the scaling constraints for a specific component.
170             </xs:documentation>
171         </xs:annotation>
172         <xs:sequence>
173             <xs:element name="LowerBound" type="xs:int"/>
174             <xs:element name="UpperBound" type="xs:int"/>
175             <xs:element name="Initial" type="xs:int"/>
176         </xs:sequence>
177     </xs:complexType>
178     <!--
179        Definition of OPTIMIS TREC parameters.
180     -->
181     <xs:element name="TrustSection" type="opt:TrustSectionType"/>
182     <xs:element name="RiskSection" type="opt:RiskSectionType"/>
183     <xs:element name="EcoEfficiencySection" type="opt:EcoEfficiencySectionType"/>
184     <xs:element name="CostSection" type="opt:CostSectionType"/>
185     <xs:element name="PriceComponent" type="opt:PriceComponentType"/>
186
187     <xs:complexType name="TRECSectionType">
188         <xs:sequence>
189             <xs:element ref="opt:TrustSection" minOccurs="0" maxOccurs="unbounded"/
                    >
190             <xs:element ref="opt:RiskSection" minOccurs="0" maxOccurs="unbounded"/>
```

```
191             <xs:element ref="opt:EcoEfficiencySection" minOccurs="0" maxOccurs="
                    unbounded"/>
192             <xs:element ref="opt:CostSection" minOccurs="0" maxOccurs="unbounded"/>
193         </xs:sequence>
194     </xs:complexType>
195     <xs:complexType name="TrustSectionType">
196         <xs:annotation>
197             <xs:documentation>
198                 Specifies the OPTIMIS trust parameters in a TREC section.
199             </xs:documentation>
200         </xs:annotation>
201         <xs:complexContent>
202             <xs:extension base="opt:ScopedSectionType">
203                 <xs:sequence>
204                     <xs:element name="MinimumTrustLevel" type="opt:TrustLevelType"
                                minOccurs="1"/>
205                     <xs:element name="SocialNetworkingTrustLevel" type="
                                opt:TrustLevelType" minOccurs="0"/>
206                     <xs:element name="TrustLevel" type="opt:TrustLevelType"
                                minOccurs="0"/>
207                     <xs:any namespace="##other" processContents="strict" minOccurs=
                                "0"
208                         maxOccurs="unbounded"/>
209                 </xs:sequence>
210             </xs:extension>
211         </xs:complexContent>
212     </xs:complexType>
213     <xs:simpleType name="TrustLevelType">
214         <xs:annotation>
215             <xs:documentation>
216                 Specifies the OPTIMIS Trust Level that is used for delegation in a
217                 federated cloud scenario.
218
219                 TODO: is there a specification of the different Trust Levels in
                        OPTIMIS?
220             </xs:documentation>
221         </xs:annotation>
222         <xs:restriction base="xs:int">
223             <xs:minInclusive value="0"/>
224         </xs:restriction>
225     </xs:simpleType>
226     <!--
227         Definition of OPTIMIS Risk Constraints.
228     -->
229     <xs:complexType name="RiskSectionType">
230         <xs:complexContent>
231             <xs:extension base="opt:ScopedSectionType">
232                 <xs:sequence>
233                     <xs:element name="RiskLevel" type="opt:RiskLevelType" minOccurs
                                ="0"/>
234                     <xs:element name="AvailabilityArray" type="
                                opt:AvailabilityArrayType"
235                             minOccurs="0"/>
236                     <xs:any namespace="##other" processContents="strict" minOccurs=
                                "0"
237                         maxOccurs="unbounded"/>
238                 </xs:sequence>
239             </xs:extension>
240         </xs:complexContent>
241     </xs:complexType>
242     <xs:complexType name="AvailabilityArrayType">
243         <xs:sequence>
244             <xs:element name="Availability" type="opt:AvailabilityType" minOccurs="
                    0"
245                     maxOccurs="unbounded"/>
246         </xs:sequence>
247     </xs:complexType>
248     <xs:complexType name="AvailabilityType">
249         <xs:simpleContent>
250             <xs:extension base="xs:double">
251                 <xs:attribute name="assessmentInterval" type="xs:duration"/>
252             </xs:extension>
253         </xs:simpleContent>
254     </xs:complexType>
```

```
255     <xs:simpleType name="RiskLevelType">
256         <xs:annotation>
257             <xs:documentation>
258                 Specifies the OPTIMIS Risk Level that is used for delegation in a
259                 federated cloud scenario.
260 
261                 TODO: see comment TrustLevelType.
262             </xs:documentation>
263         </xs:annotation>
264         <xs:restriction base="xs:int">
265             <xs:minInclusive value="0"/>
266         </xs:restriction>
267     </xs:simpleType>
268 
269     <!--
270        Definition of OPTIMIS EcoEfficiency Constraints.
271     -->
272     <xs:complexType name="EcoEfficiencySectionType">
273         <xs:complexContent>
274             <xs:extension base="opt:ScopedSectionType">
275                 <xs:sequence>
276                     <xs:element name="LEEDCertification" type="
                                opt:LEEDCertificationConstraintType" default="NotRequired"
                                />
277                     <xs:element name="BREEAMCertification" type="
                                opt:BREEAMCertificationConstraintType" default="
                                NotRequired"/>
278                     <xs:element name="EuCoCCompliant" type="xs:boolean" default="
                                false"/>
279                     <xs:element name="EnergyStarRating" type="
                                opt:EnergyStarRatingType" default="No"/>
280                   <xs:element name="ISO14000" type="opt:ISO14000Type" default="No"/
                          >
281                   <xs:element name="GreenStar" type="opt:GreenStarType" default="No
                          "/>
282                   <xs:element name="CASBEE" type="opt:CASBEEType" default="No"/>
283                   <xs:element name="EcoMetricArray" type="opt:EcoMetricArrayType"
                            minOccurs="0"/>
284                     <xs:any namespace="##other" processContents="strict" minOccurs=
                            "0"
285                             maxOccurs="unbounded"/>
286                 </xs:sequence>
287             </xs:extension>
288         </xs:complexContent>
289     </xs:complexType>
290     <xs:simpleType name="LEEDCertificationConstraintType">
291         <xs:restriction base="xs:string">
292             <xs:enumeration value="NotRequired"/>
293             <xs:enumeration value="Certified"/>
294             <xs:enumeration value="Silver"/>
295             <xs:enumeration value="Gold"/>
296             <xs:enumeration value="Platinum"/>
297         </xs:restriction>
298     </xs:simpleType>
299     <xs:simpleType name="BREEAMCertificationConstraintType">
300         <xs:restriction base="xs:string">
301             <xs:enumeration value="NotRequired"/>
302             <xs:enumeration value="Pass"/>
303             <xs:enumeration value="Good"/>
304             <xs:enumeration value="VeryGood"/>
305             <xs:enumeration value="Excellent"/>
306             <xs:enumeration value="Outstanding"/>
307         </xs:restriction>
308     </xs:simpleType>
309     <xs:simpleType name="EnergyStarRatingType">
310         <xs:union>
311             <xs:simpleType>
312                 <xs:restriction base="xs:string">
313                     <xs:enumeration value="No"/>
314                 </xs:restriction>
315             </xs:simpleType>
316             <xs:simpleType>
317                 <xs:restriction base="xs:int">
318                     <xs:minInclusive value="1"/>
```

```
                    <xs:maxInclusive value="100"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
    </xs:simpleType>
     <xs:simpleType name="ISO14000Type">
      <xs:restriction base="xs:string">
         <xs:enumeration value="No"/>
            <xs:enumeration value="ISO14001-Compliant"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="GreenStarType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="No"/>
            <xs:enumeration value="4"/>
            <xs:enumeration value="5"/>
            <xs:enumeration value="6"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CASBEEType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="No"/>
            <xs:enumeration value="C"/>
            <xs:enumeration value="B-"/>
            <xs:enumeration value="B+"/>
            <xs:enumeration value="A"/>
            <xs:enumeration value="S"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="EcoMetricArrayType">
        <xs:sequence>
            <xs:element name="EcoMetric" type="opt:EcoMetricType" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="EcoMetricType">
        <xs:annotation>
         <xs:documentation>
            Two threasholds: EnergyEfficiency and EcologicalEfficiency
            Unit: Euro/(CU/W) as Computing Units per Watt
            Unit: Euro/(CU/kgCO2) as CO2 emissions
            Unit: Euro/s as money paid for each second
            we want to establish the minimum thresholds per service in terms of
                energy efficiency
            (performance/W, currently as Computing Units per Watt, CU/W) and
                ecological efficiency
            (performance/CO2 emissions). These thresholds should be specified with
                their corresponding penalizations
            (SLA) in case they are surpassed. These penalizations need to be
                specified in terms of
            magnitude (money paid per each CU/W below the threshold) and time
            (money paid per each second the threshold is surpassed).
            In addition, they could be specified as soft (desirable thresholds,
            no penalization if surpassed) or hard (penalization if surpassed)
                thresholds.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="ThresholdValue" type="opt:ThresholdValueType" default="
             NotSpecified"/>
         <xs:element name="SLAType" type="opt:SLATypeEnum" default="Soft"/>
         <xs:element name="MagnitudePenalty" type="opt:MagnitudePenaltyType"
             default="NA"/>
         <xs:element name="TimePenalty" type="opt:TimePenaltyType" default="NA"/>
      </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="ThresholdValueType">
      <xs:union>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="NotSpecified"/>
            </xs:restriction>
         </xs:simpleType>
```

```
385           <xs:simpleType>
386              <xs:restriction base="xs:float">
387                 <xs:minInclusive value="0"/>
388              </xs:restriction>
389           </xs:simpleType>
390        </xs:union>
391     </xs:simpleType>
392     <xs:simpleType name="SLATypeEnum">
393        <xs:restriction base="xs:string">
394           <xs:enumeration value="Soft"/>
395           <xs:enumeration value="Hard"/>
396        </xs:restriction>
397     </xs:simpleType>
398     <xs:simpleType name="MagnitudePenaltyType">
399        <xs:union>
400           <xs:simpleType>
401              <xs:restriction base="xs:string">
402                 <xs:enumeration value="NA"/>
403              </xs:restriction>
404           </xs:simpleType>
405           <xs:simpleType>
406              <xs:restriction base="xs:float">
407                 <xs:minInclusive value="0"/>
408              </xs:restriction>
409           </xs:simpleType>
410        </xs:union>
411     </xs:simpleType>
412     <xs:simpleType name="TimePenaltyType">
413        <xs:union>
414           <xs:simpleType>
415              <xs:restriction base="xs:string">
416                 <xs:enumeration value="NA"/>
417              </xs:restriction>
418           </xs:simpleType>
419           <xs:simpleType>
420              <xs:restriction base="xs:float">
421                 <xs:minInclusive value="0"/>
422              </xs:restriction>
423           </xs:simpleType>
424        </xs:union>
425     </xs:simpleType>
426
427     <!--
428        Definition of OPTIMIS Cost constraints.
429     -->
430     <xs:complexType name="CostSectionType">
431        <xs:complexContent>
432           <xs:extension base="opt:ScopedSectionType">
433              <xs:sequence>
434                 <xs:element name="PricePlan" maxOccurs="unbounded"
435                             type="opt:PricePlanType" minOccurs="0">
436                 </xs:element>
437              </xs:sequence>
438           </xs:extension>
439        </xs:complexContent>
440     </xs:complexType>
441     <xs:complexType name="PricePlanType">
442        <xs:annotation>
443           <xs:documentation>
444              A PricePlan is a set of charges associated with a network-
                    provisioned
445              entity. Alternative sets of fees (i.e. alternative PricePlans) of
                    the
446              same service provision may be made available for the consumer to
                    choose
447              from, for example to offer the consumer the choice between a flat
                    price
448              scheme and a usage-based scheme (a common practice in the
449              telecommunication industry). Several PricePlans may exist for the
                    same
450              service in order to suit different user profiles and charge them
451              appropriately (e.g. heavy- and light-usage users), or as a key
                    price
452              customization instrument to individually match diverse service
```

```
453                 valuations. There are three attributes associated with the
                        PricePlan
454                 term:
455
456                 1. currency, as a name string, EString: the currency for all price
457                 amounts within this PricePlan, e.g. EUR.
458
459                 2. planCap, as a float num., EFloat: providing this maximum
                        PricePlan
460                 value prevents from charging the user a higher total price,
                        regardless
461                 of the cumulative total price the components and adjustments within
                         this
462                 PricePlan may eventually amount to. Example: A cap may be used to
                        set an
463                 upper limit in a strictly usage-based plan.
464
465                 3. planFloor, as a float num., EFloat: providing this minimum
                        PricePlan
466                 value prevents from charging the user a lower total price,
                        regardless of
467                 the cumulative total price the components and adjustments within
                        this
468                 PricePlan may eventually amount to. Example: A floor may be used to
                         set
469                 a lower limit to discounts that may result in an excessively low
                        price.
470             </xs:documentation>
471         </xs:annotation>
472
473         <xs:sequence>
474             <xs:element ref="opt:PriceComponent" maxOccurs="unbounded" minOccurs="0
                    "/>
475         </xs:sequence>
476         <xs:attribute name="planCap" type="xs:float"/>
477         <xs:attribute name="planFloor" type="xs:float"/>
478         <xs:attribute name="currency" type="xs:string"/>
479     </xs:complexType>
480
481     <xs:complexType name="PriceComponentType">
482         <xs:annotation>
483             <xs:documentation>
484                 PriceComponents are fees included in a PricePlan, which subject to
485                 conditions (expressed as PriceFences) may contribute to the total
                        amount
486                 charged. Components within the same plan are summed together in
                        order to
487                 get the total amount (price of the service). Common examples of
488                 PriceComponents that may coexist in the same PricePlan are: startup
                         or
489                 membership charges (to access the service), periodic subscription
                        fees
490                 (with a certain recurrence - e.g. monthly - as long as committed to
                         by
491                 the contract), pay-per-unit charges (whose total will be
                        proportional to
492                 the metered usage), options or feature dependent charges. The final
493                 value of the component will depend on the active PriceLevel (
                        determined
494                 by the evaluation of the relative PriceFences) and the
                        PriceAdjustments
495                 that may apply (e.g. discounts). There are two attributes
                        associated
496                 with the PriceComponent term:
497
498                 1. componentCap, as a float num., EFloat: providing this maximum
499                 PriceComponent value prevents the component final price from
                        exceeding a
500                 certain amount, regardless of its levels and the parameters they
                        are
501                 indexed to. Example: A cap may be used to set an upper limit for a
502                 component whose levels vary with usage.
503
504                 2. componentFloor, as a float num., EFloat: providing this minimum
```

```
505                 PriceComponent value prevents the component final price from
                        falling
506                 below a certain amount, regardless of its levels and the parameters
                        they
507                 are indexed to. Example: A floor may be used to set a lower limit
                        for a
508                 component whose levels vary with usage.
509             </xs:documentation>
510         </xs:annotation>
511
512         <xs:sequence>
513             <xs:element name="Name" type="xs:string"/>
514             <xs:element name="PriceLevel" type="opt:PriceLevelType"
515                         maxOccurs="unbounded"/>
516
517         </xs:sequence>
518         <xs:attribute name="componentCap" type="xs:float"/>
519         <xs:attribute name="componentFloor" type="xs:float"/>
520     </xs:complexType>
521
522     <xs:complexType name="PriceLevelType">
523         <xs:annotation>
524             <xs:documentation>
525                 PriceLevel captures amounts charged by a PriceComponent. Since each
526                 PriceComponent may assume several values depending on the provider'
                        s
527                 price segmentation strategies, it is allowed to contain multiple
528                 PriceLevels. This allows shaping charged amounts according to
                        customers'
529                 behavior and aligning usage with capacity or incurred costs (just
                        like
530                 utilities do by offering different electricity rates for different
                        times
531                 of day).
532             </xs:documentation>
533         </xs:annotation>
534         <xs:sequence>
535             <xs:element name="PriceType" type="xs:string"/>
536             <xs:element name="Name" type="xs:string"/>
537             <xs:element name="AbsoluteAmount" type="xs:decimal"/>
538             <xs:element name="Multiplier" type="xs:string"/>
539         </xs:sequence>
540     </xs:complexType>
541
542     <xs:complexType name="QuantityLiteralsArrayType">
543         <xs:sequence>
544             <xs:element name="Quantity" type="opt:QuantityType" maxOccurs="
                    unbounded"/>
545         </xs:sequence>
546     </xs:complexType>
547
548     <xs:complexType name="QuantityType">
549         <xs:sequence>
550             <xs:element name="Amount" type="xs:decimal"/>
551             <xs:element name="TypeReference" type="xs:string"/>
552         </xs:sequence>
553         <xs:attribute name="id" type="xs:string" use="required"/>
554     </xs:complexType>
555
556     <!--
557
558         Definition of the ElasticityArray. The definition of the RuleType is based
                on the
559         Reservoir Elasticity Array. (see schema: http://schemas.telefonica.com/
                claudia/ovf)
560     -->
561     <xs:element name="ElasticityRule" type="opt:ElasticityRuleType"/>
562
563     <xs:complexType name="ElasticitySectionType">
564         <xs:choice>
565             <xs:sequence>
566                 <xs:element name="SPManagedElasticity" nillable="true"/>
567             </xs:sequence>
568             <xs:sequence>
```

```
569                 <xs:element name="VariableSet" type="opt:VariableSetType"/>
570                 <xs:element name="ElasticityRules">
571                     <xs:complexType>
572                         <xs:sequence>
573                             <xs:element ref="opt:ElasticityRule" maxOccurs="
                                    unbounded"/>
574                         </xs:sequence>
575                     </xs:complexType>
576                 </xs:element>
577             </xs:sequence>
578         </xs:choice>
579     </xs:complexType>
580
581     <xs:complexType name="ElasticityRuleType">
582         <xs:complexContent>
583             <xs:extension base="opt:ScopedSectionType">
584                 <xs:sequence>
585                     <xs:element name="Condition" type="opt:ConditionType"/>
586                     <xs:element name="Effect" type="opt:Effect"/>
587                 </xs:sequence>
588                 <xs:attribute name="name" type="xs:string" use="required"/>
589             </xs:extension>
590         </xs:complexContent>
591     </xs:complexType>
592
593     <xs:complexType name="ConditionType">
594         <xs:sequence>
595             <xs:element name="Expression" type="xs:string"/>
596             <xs:element name="AssessmentCriteria" type="opt:AssessmentCriteria"/>
597         </xs:sequence>
598     </xs:complexType>
599
600     <xs:complexType name="AssessmentCriteria">
601         <xs:sequence>
602             <xs:element name="Window" type="xs:duration"/>
603             <xs:element name="Frequency" type="xs:int"/>
604         </xs:sequence>
605     </xs:complexType>
606
607     <xs:complexType name="Effect">
608         <xs:sequence>
609             <xs:element name="Importance" type="xs:int"/>
610             <xs:element name="Action" type="xs:string"/>
611         </xs:sequence>
612     </xs:complexType>
613
614     <xs:element name="Variable" type="opt:VariableType"/>
615
616     <xs:complexType name="VariableSetType">
617         <xs:sequence>
618             <xs:element ref="opt:Variable" maxOccurs="unbounded"/>
619         </xs:sequence>
620     </xs:complexType>
621
622     <xs:complexType name="VariableType">
623         <xs:annotation>
624             <xs:documentation>
625                 Location: the path to a location where the value can be found.
626                 The type attribute [internal | external] specifies if the variable
                        can be found internal in the
627                 manifest itself (location would be an xpath expression) or at some
                        external
628                 location, e.g URL to REST address of a monitoring system which
                        provides the current value of
629                 the variable.
630                 The metric attribute specifies the type of the value received at
                        location, eg. int
631                 The name specifies the variable name which will be used in the
                        rules.
632             </xs:documentation>
633         </xs:annotation>
634         <xs:sequence>
635             <xs:element name="Location" type="xs:string"/>
636         </xs:sequence>
```

```
637         <xs:attribute name="name" type="xs:string" use="required"/>
638         <xs:attribute name="metric" type="xs:string"/>
639         <xs:attribute name="type" type="opt:ElasticityLocationTypeEnum" use="
                required"/>
640     </xs:complexType>
641
642
643     <xs:simpleType name="ElasticityLocationTypeEnum">
644         <xs:restriction base="xs:string">
645             <xs:enumeration value="internal"/>
646             <xs:enumeration value="external"/>
647         </xs:restriction>
648     </xs:simpleType>
649
650
651     <xs:complexType name="ScopeArrayType">
652         <xs:sequence>
653             <xs:element name="ComponentId" type="xs:string" maxOccurs="unbounded"/>
654         </xs:sequence>
655     </xs:complexType>
656     <xs:simpleType name="PositiveDecimalType">
657         <xs:restriction base="xs:decimal">
658             <xs:minExclusive value="0"/>
659         </xs:restriction>
660     </xs:simpleType>
661     <!--
662        Definition of the OPTIMIS data protection constraints.
663     -->
664     <xs:complexType name="DataProtectionSectionType">
665         <xs:sequence>
666             <xs:element name="EligibleCountryList" type="opt:CountryListType"
667                         minOccurs="0"/>
668             <xs:element name="NonEligibleCountryList" type="opt:CountryListType"
669                         minOccurs="0"/>
670             <xs:element name="DataProtectionLevel" type="
                    opt:DataProtectionLevelType"
671                         minOccurs="0"/>
672             <xs:element name="DataEncryptionLevel" type="opt:EncryptionLevelType"
673                         minOccurs="0"/>
674             <xs:element ref="opt:DataStorage" minOccurs="0" maxOccurs="unbounded"/>
675             <!--  <xs:element name="DataStorage" type="opt:DataStorageType"
                    minOccurs="0" maxOccurs="unbounded"/>  -->
676
677             <xs:element name="SCC" type="opt:SCCType" minOccurs="0" maxOccurs="1"/>
678             <xs:element name="BCR" type="opt:BCRType" minOccurs="0" maxOccurs="1"/>
679             <xs:element name="IPR" type="opt:IPRType" minOccurs="0" maxOccurs="1"/>
680
681             <xs:any namespace="##other" processContents="strict" minOccurs="0"
682                     maxOccurs="unbounded"/>
683         </xs:sequence>
684     </xs:complexType>
685     <xs:element name="DataStorage" type="opt:DataStorageType"/>
686     <xs:complexType name="DataStorageType">
687         <xs:complexContent>
688             <xs:extension base="opt:ScopedSectionType">
689                 <xs:sequence>
690                     <xs:element name="Name" type="xs:string" default="storage-name"
                            />
691                     <xs:element name="AllocationUnit" type="xs:string" default="
                            byte"/>
692                     <xs:element name="Capacity" type="xs:long" default="1"/>
693                 </xs:sequence>
694             </xs:extension>
695         </xs:complexContent>
696     </xs:complexType>
697     <xs:simpleType name="DataProtectionLevelType">
698         <xs:annotation>
699             <xs:documentation>
700                 DataProtectionLevel specifies the level of protection that is
                        guaranteed
701                 by a service provider regarding data management. In general it
                        defines
702                 to which countries data may be transfered by the provider.
                        Countries are
```

```
703                 divided into countries that have a sufficient level of protection (
                        known
704                 as Data Protection Area–DPA) and countries that do not meet these
705                 levels. Transferring sensitive data to the latter is a violation
                        and the
706                 cloud providers engaged in federations should have the necessary
707                 framework to prevent this from happening. By law, the Cloud
                        Provider
708                 does not have the obligation to keep the data in one particular
                        country
709                 of the DPA. The DataProtectionLevelType specifies whether the data
710                 included in the service under consideration is sensitive or not. If
                         not,
711                 there are no limitations to their transfer. If yes, they should be
712                 restricted to countries that are part of the DPA. The list of the
                        DPA
713                 countries is the following:
714
715                 – all 27 EU Member States – all countries of the European Economic
                        Area
716                 (Iceland, Liechtenstein, Norway) – Switzerland – Canada – Argentina
                         –
717                 Guernsey – Isle of Man – US organisations who take part in the US
                        safe
718                 harbour program – And the state of Israel.
719
720                 TODO: Is there a maintained reference list of DPA countries online
721                 available?
722             </xs:documentation>
723         </xs:annotation>
724         <xs:restriction base="xs:string">
725             <xs:enumeration value="DPA"/>
726             <xs:enumeration value="None"/>
727         </xs:restriction>
728     </xs:simpleType>
729     <xs:simpleType name="ISO3166Alpha2">
730         <xs:annotation>
731             <xs:documentation>
732                 Two-letter (alpha-2) ISO 3166-1 code for one of the 243 countries.
                        These
733                 codes are subject to change. For valid values refer to
734                 http://www.iso.org/iso/list-en1-semic-3.txt
735             </xs:documentation>
736         </xs:annotation>
737         <xs:restriction base="xs:string">
738             <xs:whiteSpace value="collapse"/>
739             <xs:pattern value="[A-Z]{2}"/>
740         </xs:restriction>
741     </xs:simpleType>
742     <xs:complexType name="CountryListType">
743         <xs:sequence>
744             <xs:element name="Country" type="opt:ISO3166Alpha2" maxOccurs="
                    unbounded"/>
745         </xs:sequence>
746     </xs:complexType>
747     <xs:complexType name="EncryptionLevelType">
748         <xs:choice>
749             <xs:sequence>
750                 <xs:element name="EncryptionAlgoritm"
751                             type="opt:EncryptionAlgoritmType"/>
752                 <xs:element name="EncryptionKeySize" type="xs:int" default="128"
753                             minOccurs="0"/>
754             </xs:sequence>
755             <xs:sequence>
756                 <xs:element name="CustomEncryptionLevel" type="xs:anyType"/>
757             </xs:sequence>
758         </xs:choice>
759     </xs:complexType>
760     <xs:simpleType name="EncryptionAlgoritmType">
761         <xs:restriction base="xs:string">
762             <xs:enumeration value="NotApplicable"/>
763             <xs:enumeration value="AES"/>
764             <xs:enumeration value="Twofish"/>
765             <xs:enumeration value="AES-Twofish"/>
```

```
766             <xs:enumeration value="AES-Twofish-Serpent"/>
767             <xs:enumeration value="Serpent-AES"/>
768             <xs:enumeration value="Serpent-Twofish-AES"/>
769             <xs:enumeration value="Twofish-Serpent"/>
770         </xs:restriction>
771     </xs:simpleType>
772     <xs:complexType name="SCCType">
773       <xs:annotation>
774             <xs:documentation>
775                 Standard Contractual Clauses
776             </xs:documentation>
777         </xs:annotation>
778         <xs:sequence>
779             <xs:element name="apply" type="xs:boolean" default="false"/>
780             <xs:element name="Location" type="xs:string" minOccurs="0" maxOccurs="1
                    "/>
781             <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs
                    ="1"/>
782             <xs:element name="Clause" type="opt:SectionType" minOccurs="0"
                    maxOccurs="unbounded"/>
783         </xs:sequence>
784     </xs:complexType>
785     <xs:complexType name="BCRType">
786       <xs:annotation>
787             <xs:documentation>
788                 Binding Coorporate Rules
789             </xs:documentation>
790         </xs:annotation>
791         <xs:sequence>
792             <xs:element name="apply" type="xs:boolean" default="false"/>
793             <xs:element name="Location" type="xs:string" minOccurs="0" maxOccurs="1
                    "/>
794             <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs
                    ="1"/>
795             <xs:element name="Rule" type="opt:SectionType" minOccurs="0" maxOccurs=
                    "unbounded"/>
796         </xs:sequence>
797     </xs:complexType>
798     <xs:complexType name="IPRType">
799       <xs:annotation>
800             <xs:documentation>
801                 Intellectual Property Rights
802             </xs:documentation>
803         </xs:annotation>
804         <xs:sequence>
805             <xs:element name="apply" type="xs:boolean" default="false"/>
806             <xs:element name="Location" type="xs:string" minOccurs="0" maxOccurs="1
                    "/>
807             <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs
                    ="1"/>
808             <xs:element name="Rule" type="opt:SectionType" minOccurs="0" maxOccurs=
                    "unbounded"/>
809         </xs:sequence>
810     </xs:complexType>
811     <xs:complexType name="SectionType">
812         <xs:sequence>
813             <xs:element name="Title" type="xs:string" minOccurs="0" maxOccurs="1"/>
814             <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs
                    ="1"/>
815             <xs:element name="Item" type="xs:string" minOccurs="0" maxOccurs="
                    unbounded"/>
816         </xs:sequence>
817     </xs:complexType>
818
819     <!--
820         Definition of the AffinitySection.
821     -->
822     <xs:element name="AffinityRule" type="opt:AffinityRuleType"/>
823     <xs:complexType name="AffinitySectionType">
824         <xs:sequence>
825             <xs:element ref="opt:AffinityRule" maxOccurs="unbounded" minOccurs="0"/
                    >
826         </xs:sequence>
827     </xs:complexType>
```

```
828     <xs:complexType name="AffinityRuleType">
829         <xs:complexContent>
830             <xs:extension base="opt:ScopedSectionType">
831                 <xs:sequence>
832                     <xs:element name="AffinityConstraints" type="
                            opt:AffinityConstraintType"/>
833                 </xs:sequence>
834             </xs:extension>
835         </xs:complexContent>
836     </xs:complexType>
837     <xs:simpleType name="AffinityConstraintType">
838         <xs:restriction base="xs:string">
839             <xs:enumeration value="High"/>
840             <xs:enumeration value="Medium"/>
841             <xs:enumeration value="Low"/>
842         </xs:restriction>
843     </xs:simpleType>
844 
845     <!--
846        Definition of the Anti AffinitySection.
847     -->
848     <xs:element name="AntiAffinityRule" type="opt:AntiAffinityRuleType"/>
849     <xs:complexType name="AntiAffinitySectionType">
850         <xs:sequence>
851             <xs:element ref="opt:AntiAffinityRule" maxOccurs="unbounded" minOccurs=
                    "0"/>
852         </xs:sequence>
853     </xs:complexType>
854     <xs:complexType name="AntiAffinityRuleType">
855         <xs:complexContent>
856             <xs:extension base="opt:ScopedSectionType">
857                 <xs:sequence>
858                     <xs:element name="AntiAffinityConstraints" type="
                            opt:AntiAffinityConstraintType"/>
859                 </xs:sequence>
860             </xs:extension>
861         </xs:complexContent>
862     </xs:complexType>
863     <xs:simpleType name="AntiAffinityConstraintType">
864         <xs:restriction base="xs:string">
865             <xs:enumeration value="High"/>
866             <xs:enumeration value="Medium"/>
867             <xs:enumeration value="Low"/>
868         </xs:restriction>
869     </xs:simpleType>
870 
871     <xs:complexType name="ServiceEndpointsType">
872         <xs:sequence>
873             <xs:element name="ServiceEndpoint" minOccurs="0" maxOccurs="unbounded"
                    type="opt:ServiceEndPointType">
874 
875             </xs:element>
876         </xs:sequence>
877     </xs:complexType>
878 
879     <xs:complexType name="ServiceEndPointType">
880         <xs:simpleContent>
881             <xs:extension base="xs:anyURI">
882                 <xs:attribute name="name" type="xs:string"/>
883             </xs:extension>
884         </xs:simpleContent>
885     </xs:complexType>
886 </xs:schema>
```

A.3 Curriculum Vitae

Personal Data

Name:	Wolfgang Ziegler
Date of birth:	09.08.1953
Nationality:	German
Email:	wolfgang.ziegler@scai.fraunhofer.de

Education

2015:	Doctoral student in Computer Science
	Institute of Computer Science
	Georg-August-Universität Göttingen
2013:	Diploma in Computer Science
	FernUniversität in Hagen

Professional Experience

Senior Researcher
SCAI, Sankt Augustin, Germany
2002 - to date

- The focus of my research is on
  - Cloud Computing
  - Data protection and data security in the fields of Cloud Computing and Big/Smart Data
  - Dynamic machine-processable Service Level Agreement (SLA) and Service Level Management
  - Technologies for software licensing and license management in distributed computing
  - Resource management and scheduling
- Establishing and leading the research group for Grid and Cloud middleware
- Defining and establishing the SCRUM processes of the research group as certified scrum product owner
- Leading the development of elasticLM (software license management for distributed computing)

- Writing proposals for and contributing to research projects funded by Federal Ministry of Education and Research (BMBF) and Federal Ministry for Economic Affairs and Energy (BMWi)
- Writing proposals for and contributing to European funded research projects
- Reviewing of research project proposals and projects for the European Commission
- Supervising students
- Reviewing for various conferences and journals
- Contributing to standards development
  - Vice President of Community at the Open Grid Forum
  - Area Director Applications Area at the Open Grid Forum
  - Co-chair of the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG) working group at the Open Grid Forum
  - Member of ISO/IEC JTC 1 SC38 working group 3 - Cloud Computing
  - Member of two working groups of The Institute of Electrical and Electronics Engineers (IEEE): P2301 - Guide for Cloud Portability and Interoperability Profiles (CPIP) and P2302 - Intercloud

Research Associate
Gesellschaft für Mathematik und Datenverarbeitung (GMD), Sankt Augustin, Germany
1987 - 2001

- Member of the group responsible for the operation of GMD's mainframe infrastructure
- Member of the GMD hosted lab for massive parallel systems (Suprenum, CM2, CM5, Alliant, SP2) of the Höchstleistungsrechenzentrum (HLRZ)
- Planning and installation of GMD's visualisation lab
- Conducting co-operative research projects with NEC (operation of Cenju systems, further development of a scheduling system for massive parallel computers)

Member of the Technical Staff
Universität Mannheim, Mannheim, Germany
1983 - 1986

- Head of system administration of the Data Processing Centre at Universität Mannheim

Self-employed
1978 - 1982

- Silk screen printing shop

Student Assistant
1973 - 1977

- Software development and consulting at the Sonderforschungsbereich 24, Universität Mannheim

Teaching

Summer Schools

- CoreGRID Summer School 2006 (co-organiser and lecturer)
- CoreGRID Summer School 2008 (lecturer)
- CEA-EDF Summer School 2009 (lecturer)

Bonn-Aachen International Center for Information Technology (B-IT)

- Seminar Introduction to Grid Computing, Winter Semester 2006/2007
- Seminar Introduction to Life Science Grid Computing, Winter Semester 2007/2008

Supervision

Khan, M.Z.: Securing medical data inside Cloud using cloud4health as use case.
Universität Bonn, 2015. (Master)

Bolenz, P.: Eine Benutzeroberfläche zur Darstellung und Manipulation komplexer XML-Daten von Service Level Agreements.
Hochschule Bonn-Rhein-Sieg, 2014. (Bachelor)

Reiser, S.: A XML-based benchmark toolset for simple creation of integration tests for a specific C and JAVA API.
Hochschule Bonn-Rhein-Sieg, 2014. (Bachelor)

Pauksztelo, D.: Design and Development of a Graphical User Interface for a Command Line-based License Management System.
Hochschule Bonn-Rhein-Sieg, 2013. (Bachelor)

Mohebbi, P.: Development and implementation of a secure gateway for processing clinical data in a health cloud.
Technische Hochschule Köln, 2013. (Master)

Weuffel, T.: Vorbelegung und lose Reservierung von Ressourcen in lokalen Cluster-Umgebungen : Evaluation, Modellierung und Implementierung eines objektorientierten Cluster-Schedulers unter Berücksichtigung von Advance Reservation und loser Reservierung.
Hochschule Bonn-Rhein-Sieg, 2009. (Master)

Faroughi, A.: Integration von VO-Management Technologien in UNICORE.
Technische Hochschule Köln, 2007. (Master)

Faroughi, R.: Integration von VO-Management Technologien in UNICORE.
Technische Hochschule Köln, 2007. (Master)

Shahid, M.: Large scale virtual high throughput screening (vHTS) on an optical high speed testbed.
Bonn, Bonn-Aachen International Center for Information Technology (B-IT), 2007. (Master)

Weuffel, T.: Anbindung des MetaScheduling Service an die UNICORE 5 Resource Broker Schnittstelle.
Hochschule Bonn-Rhein-Sieg, 2007. (Bachelor)

Wäldrich, O.: Meta-Scheduling-Architekturen in komplexen Grid-Umgebungen.
Hochschule Bonn-Rhein-Sieg, 2005. (Master)

Publications

For complete list of publications browse at
http://www.scai.fraunhofer.de/en/about-us/staff/ziegler.html#tabpanel-2

or have a look into the Fraunhofer publication database at
http://publica.fraunhofer.de/starweb/pub09/en/index.htm.