

## Reliable Orchestration of Resources using WS-Agreement

*Heiko Ludwig*

*hludwig@us.ibm.com*

*IBM TJ Watson Research Center, Hawthorne, NY, USA*

*Toshiyuki Nakata*

*t-nakata@cw.jp.nec.com*

*NEC, Central Research Laboratory, Tokyo, Japan*

*Philipp Wieder*

*ph.wieder@fz-juelich.de*

*Research Centre Jülich, 52425 Jülich, Germany*

*Oliver Wäldrich*

*Wolfgang Ziegler*

*{oliver.waeldrich/wolfgang.ziegler}@scai.fraunhofer.de*

*Fraunhofer Institute SCAI, 53754 Sankt Augustin, Germany*

---



CoreGRID Technical Report

Number TR-0050

October 5, 2006

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence

URL: <http://www.coregrid.net>

# Reliable Orchestration of Resources using WS-Agreement

Heiko Ludwig

hludwig@us.ibm.com

IBM TJ Watson Research Center, Hawthorne, NY, USA

Toshiyuki Nakata

t-nakata@cw.jp.nec.com

NEC, Central Research Laboratory, Tokyo, Japan

Philipp Wieder

ph.wieder@fz-juelich.de

Research Centre Jülich, 52425 Jülich, Germany

Oliver Wäldrich

Wolfgang Ziegler

{oliver.waeldrich|wolfgang.ziegler}@scai.fraunhofer.de

Fraunhofer Institute SCAI, 53754 Sankt Augustin, Germany

*CoreGRID TR-0050*

October 5, 2006

## Abstract

Co-ordinated usage of resources in a Grid environment is a challenging task impeded by the nature of resource usage and provision: Resources reside in different geographic locations, are managed by different organisations, and the provision of reliable access to these resource usually has to be negotiated and agreed upon in advance. These prerequisites have to be taken into account providing solutions for the orchestration of Grid resources. In this document we describe the use of WS-Agreement for Service Level Agreements paving the way for using multiple distributed resources to satisfy a single service request. WS-Agreement is about to be released as a draft recommendation of the Global Grid Forum and has already been implemented in a number of projects, two of which we will presented in this paper.

## 1 Introduction

### 1.1 Motivation to use WS-Agreement

Today, the access to and use of services offered by a provider, e.g. use of resources in a Grid environment, are usually governed by static agreements made beforehand between the parties. The parties thus already have to be in contact before the actual (business) interaction takes place. However, in a large Grid environment with a large number of providers of different services and an even larger number of service consumers this static approach is not feasible.

In contrast WS-Agreement [1] offers a reliable mechanism to create solid electronic agreements between different parties interested in setting up dynamic collaborations that include mutual obligations, e.g. between service provider

---

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

and service consumer. WS-Agreement thus fills the existing gap when trying to establish a dependable relation between these parties (that must not have established a formal contractual relationship beforehand) enabling them e.g. to provide a service on one side and to consume this service at the other side while the formal framework of this exchange is governed by a dynamic contract. WS-Agreement empowers both parties to establish a dynamic electronic contract that governs their ad hoc business connection set up to meet an actual need of one party with a service provided by the other party. Once both sides have fulfilled their obligations specified in the agreement, the agreement is completed and both partners have no longer mutual obligations from this agreement.

WS-Agreement is now in the state of a proposed recommendation which is the last phase before becoming a recommendation of the Global Grid Forum (GGF [5]). Nevertheless, it is already used in a number of projects - two of them being presented in this paper - and has attracted attention from several large communities like the telecommunications and IT industries organised in the IPsphere Forum [7] or the partners in the European project EGEE [4].

## 1.2 WS-Agreement

The objective of the WS-Agreement draft specification, defined by the GRAAP Working Group [6] of the GGF, is to provide a domain-independent and standard way to establish and monitor Service Level Agreements. The specification comprises three major elements: (i) A description format for agreement templates and agreements; (ii) a basic protocol for establishing agreements, and (iii) an interface specification to monitor agreements at runtime.

Agreements according to the specification are bilateral and set up between two roles, the *Agreement Initiator* and the *Agreement Responder*. These roles are independent of the roles of service provider, the domain that performs jobs or other services, and service consumer. An agreement defines a dynamically-established and dynamically-managed relationship between these parties. The object of the relationship is the delivery of a service, e.g. the execution of a job or the availability of compute resources, by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only functional properties for identification or creation of the service, but also non-functional properties of the service such as performance or availability.

Fig. 1 outlines the main concepts and interfaces of the WS-Agreement specification. In the chosen example, the Agreement Responder is a service provider, the Agreement Initiator the service consumer. An Agreement Responder exposes an interface of an *Agreement Factory*, which offers an operation to create an agreement and an operation to retrieve a set of agreement templates proposed by the agreement provider. Agreement templates are agreements with fields to be filled in. Templates help an Agreement Initiator to create agreements that the agreement provider can understand and accept. The `createAgreement` operation returns `accept` or `reject`, if a synchronous reply is expected. Otherwise, in case of a longer decision-making process, the service responder can convey the decision to an `AgreementResponse` interface that the Initiator exposes. If the `createAgreement` operation succeeds, an agreement instance is created. The agreement instance exposes the terms of the agreement as properties that can be queried. In addition, runtime states for the agreement as a whole and its individual terms can be inspected by the Initiator. All interfaces exposed by the parties, *Agreement Factory*, *Agreement* and *AgreementResponse* are resources according to the Web Services Resource Framework (WSRF) [13]. Upon acceptance of an agreement, both service provider and service consumer have to prepare for the service, which typically depends on the kind of service subject to the agreement. For example, a service provider schedules a job that is defined in the agreement. A service consumer will make the stage-in files available as defined in the agreement. Further service specific interaction may take place between the parties governed by the agreement.

The WS-Agreement draft specification defines the content model of both, agreements and agreement templates as an XML-based language. Structurally, an agreement consists of a name, a context section, and the agreement terms. The agreement context contains definitorial content such as the definition of the parties and their roles in the agreement. The agreement terms represent contractual obligations and include a description of the service as well as the specific guarantees given. A service description term (SDT) can be a reference to an existing service, a domain specific description of a service (e.g. a job using the Job Submission Description Language (JSDL [2], a data service using Data Access and Integration Services, etc.), or a set of observable properties of the service. Multiple SDTs can describe different aspects of the same service. A guarantee term on the other hand specifies non-functional characteristics in service level objectives as an expression over properties of the service, an optional qualifying condition under which objectives are to be met, and an associated business value specifying the importance of meeting these objectives.

The WS-Agreement specification only defines the top-level structure of agreements and agreement templates. This outer structure must be complemented by means of expressions suitable for a particular domain. For example, a

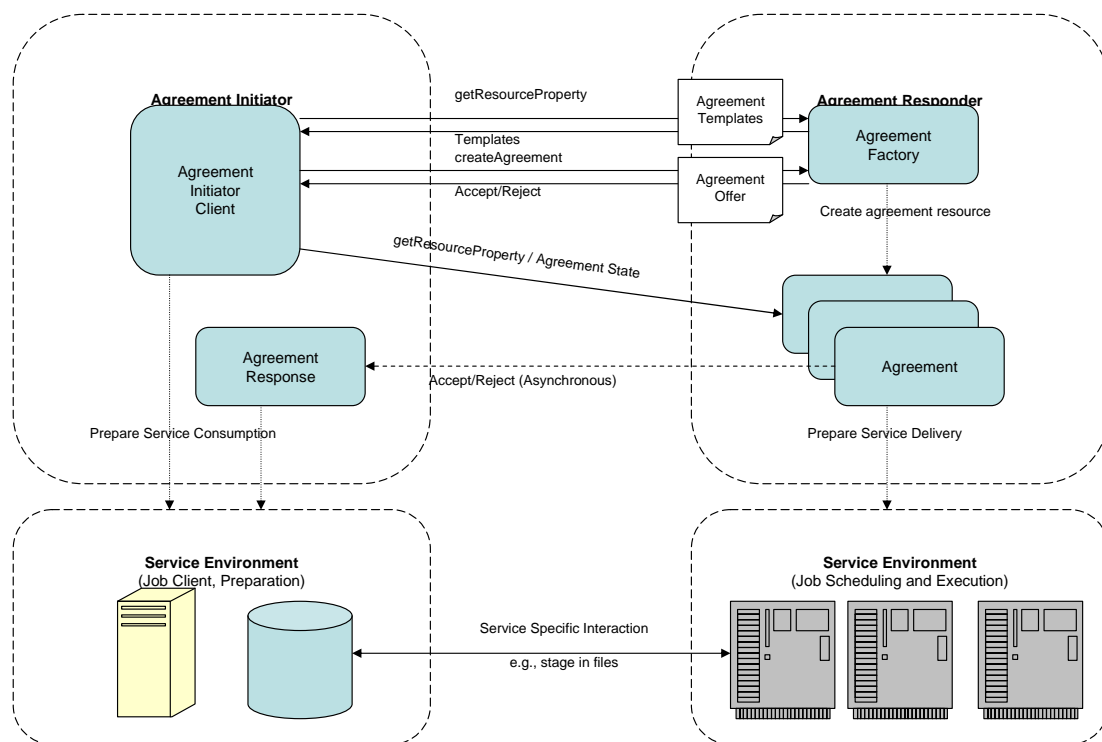


Figure 1: Concepts and interfaces of WS-Agreement

guarantee term is defined as comprising the elements scope, qualifying condition, service level objective, and business value. There are no language elements defined to specify a service level objective. Parties have to choose a suitable condition language to express the logic expressions defining a service level objective. Agreement Templates contain the same content structure as agreements but add a constraints section. This section defines which content of a template agreement can be changed by an agreement initiation and the constraints which must be met when filling in a template to create an agreement offer. A constraint comprises a named pointer to an XML element in the context or term sections of the agreement and a constraint expression defining the set of eligible values that can be filled in at this position. For example, an Initiator might be able to choose between a number of options of a job or must specify the location of a stage in file.

The remainder of the paper is organised as follows. In Section 2 we present two implementations of WS-Agreement, followed by a summary of the experiences made (Section 3). Based on these experiences we present the discussion of requirements for negotiation of Service Level Agreements in Section 4. An overview of further developments in the GRAAP-WG related to this paper concludes the paper.

## 2 Implementations

### 2.1 Wide Area Business Grid

In this section, we describe a system using WS-Agreement in combination with JSDL to express SDTs for complex, wide-area jobs. This system was developed within the framework of the Japanese Business Grid Project [3] (2003-2005). The project's mission was to develop a business Grid middleware realising a next generation business application infrastructure.

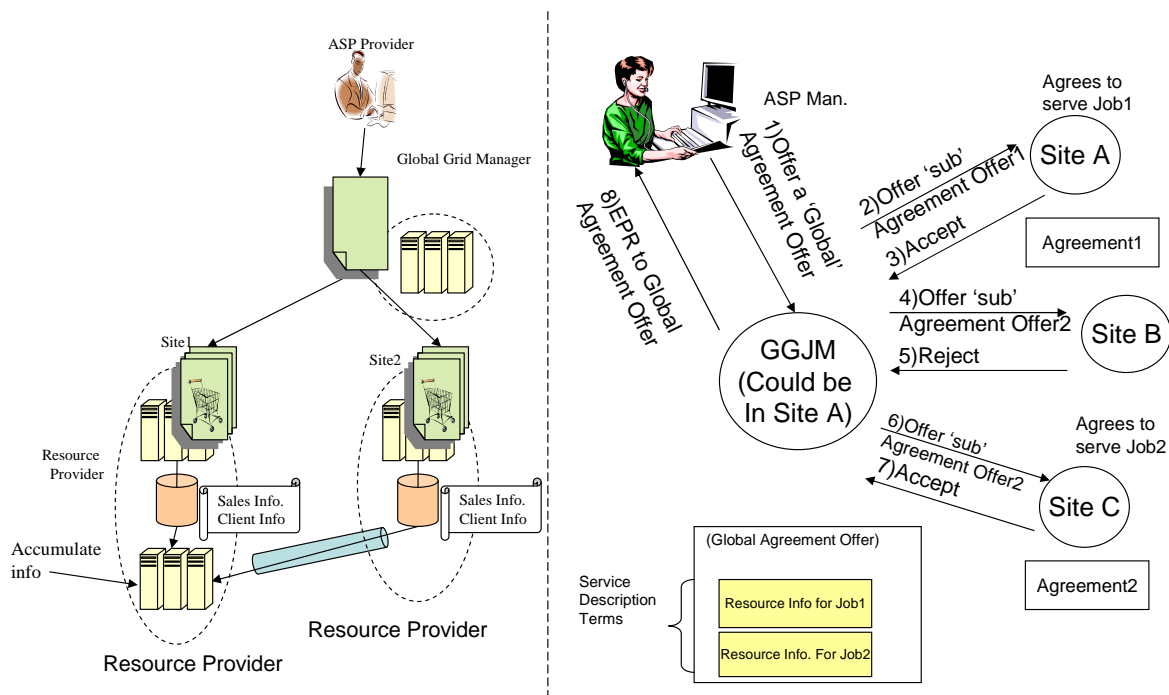


Figure 2: Wide Area Business Grid

### 2.1.1 Overview of the system

The main objective designing and implementing the system has been to make it easy to deploy and run business applications like a 3-tier Web application in a data centre. The main characteristics of the system are:

- Job submission is realised by a job description using WS-Agreement and JSDL, and the Application Contents Service (ACS).
- Brokering is used to allocate the necessary resources.
- Automatic deployment and configuration of programmes and data (including the necessary preparation of hosting environments) is realised by a language similar to the Configuration Description Language (CDL [8]), and the ACS.
- Management of heterogeneous resources is realised through Grid middleware agents providing a common interface (resource virtualisation).

### 2.1.2 Use of WS-Agreement and JSDL

We chose JSDL with extensions as the domain-specific language used within the Agreement. The link between JSDL constructs and WS-Agreement is realised through a convention specifying that the content of the JSDL element *JobName* corresponds to WS-Agreement's *ServiceName*. We also extended the JSDL constructs in order to make the description of more complicated resource requirements easier. This resulted in two types of resource description information for each job: (i) "Global" resource information for describing the type of the job or providing general characteristics of the job (e.g. whether to allow automatic load control or not) and (ii) "local" resource information

which describes the resource needed for each of the components that make up a pool to carry out the job. Several jobs which are related - such as a 3-tier Web application for a Web shop and a batch job which calculates the sales info every weekend - can be included in a single agreement.

### 2.1.3 Implementation of the Wide Area Business Grid

The agreement template as described above is utilised in order to realise a wide-area business Grid. The aim here is to make it possible to share IT resources based on the contract/agreement among (i) distributed centres in an enterprise and (ii) trusted partners' data centres (resource providers), thus making it possible for an Application Service Provider (ASP) to dispatch a complex job from a single portal, as depicted on the left side of Fig. 2. The right side of this figure pictures a scenario where an ASP wants to dispatch a complex job consisting of two 3-tier Web applications spanning over two sites (resource providers). The sequence of events which occur in this scenario is as follows:

1. The Application Service Provider's manager prepares an *Agreement Offer* which contains resource descriptions of both Job1 for one site and Job2 for another site, and sends the offer to the Global Grid Job Manager (GGJM).
2. The GGJM splits the Agreement Offer into two Agreement Offers, one for each sub-job, and offers the Agreement for Job1 to site A.
3. Site A decides to accept the Job, creates agreement1 and returns the *Endpoint Reference* (EPR) of the Agreement.
4. The GGJM stores the EPR of agreement1 internally and then offers the Agreement Offer for Job2 to site B.
5. Site B decides to reject the offer.
6. GGJM offers the Agreement Offer for Job2 to site C.
7. Site C decides to accept the offer, creates agreement2 and returns its EPR.
8. The GGJM stores the EPR of agreement2, creates an agreement for the complete job and returns it to the ASP manager.

The flexibility introduced by the combination of WS-Agreement and JSDL allows to handle very complicated jobs in a wide-area distributed environment.

## 2.2 VIOLA Meta-Scheduling Environment

The German VIOLA project [11] develops among other components a meta-scheduling environment providing resource reservations based on WS-Agreement. The immediate objective of this development is to co-allocate computational and network resources in a UNICORE-based Grid, but we designed the environment to support arbitrary types of resources and to be integrated into different Grid middleware systems. The main integration effort to access other middleware, like e.g. Globus, is to implement the client-side interface of the Meta-Scheduling Service. Since it is beyond the scope of this paper to explain the system in detail we refer to [12] for a complete architectural description of it and to [10] for the definition of the negotiation protocol currently implemented.

Fig. 3 sketches the basic architecture of the meta-scheduling environment and its integration into an arbitrary Grid middleware. The VIOLA Meta-Scheduling Service communicates with a client application using WS-Agreement, it receives a list of resources (pre-selected by a resource selection service which is not pictured here), and it returns reservations for some or all of these resources. To interact with varying types of scheduling systems we use the adapter pattern approach. The role of an Adapter is to provide a single interface to the Meta-Scheduling Service by encapsulating the specific interfaces of the different local scheduling systems. Thus the Meta-Scheduling Service can negotiate resource usage by exploiting a single interface independent of the underlying resource type. To achieve this, the Meta-Scheduling Service first queries local scheduling systems for the availability of the requested resources and then negotiates the reservations across all local scheduling systems which, in order to participate in the negotiation process, have to be capable and willing to

- let the Meta-Scheduling Service reserve resources in advance by offering data about job execution start and stop times, and

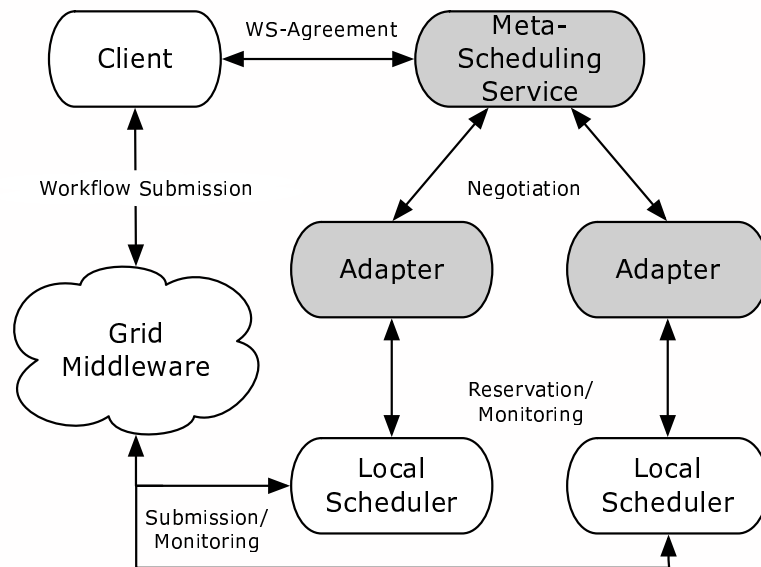


Figure 3: High-level meta-scheduling architecture

- provide at least partial access to their local schedules, e.g. by publishing information about available free time slots.

A prototype of the architecture shown in Fig. 3 has been implemented using the UNICORE Grid middleware and this prototype is used within the VIOLA testbed to demonstrate the execution of MPI jobs supported by the co-allocation capabilities of the Meta-Scheduling Service. Furthermore a network resource management system supporting advance reservation is integrated to enable the reservation and scheduling of end-to-end network connections with a guaranteed Quality of Service level.

### 3 Experience with Implementing WS-Agreement

By implementing WS-Agreement-based resource orchestration frameworks in the two different projects we gained substantial experience which we continuously feed into the standardisation process. In this section we examine some issues that we think are of value for potential implementers of WS-Agreement.

#### 3.1 Wide Area Business Grid

One of the objectives of Business Grid was to make the system dynamically adapt to workload fluctuation. Specifying a range of CPU resources together with SLA conditions using Guarantee Terms is one candidate to meet our goals, but we wanted to let the user be able to specify the controlling of the SLA in a flexible manner. To meet our goals, we use a combination of a statically allocated resource range specified by an Agreement together with policies specified by ASP managers. As a by-product, we currently do not use the Guarantee Terms. In addition, some of the domain-specific terms are specified outside the Agreement.

Implementation-specific issues worth mentioning are that the system uses J2EE RI with Pointbase for persistent storage, the Globus Toolkit 4 for WSRF-related services, and, since reserving resources on multiple sites can be a very complex and time consuming task, we use the `createPendingAgreement` operation rather than the `createAgreement` one.

#### 3.2 VIOLA Meta-Scheduling Environment

Since the client automatically generates an Agreement Offer based on the input of a user, the user cannot change explicitly the structure of an Agreement Offer. With respect to the requirements of the VIOLA target domain (MPI

jobs) this is a feasible approach, but the application to other domains may change this. An Offer specifies the required compute resources by using Service Description Terms and the required Network QoS by using Guaranty Terms. On reception of an Agreement Offer the Meta-Scheduling Service tries to co-allocate all required resources and only in case of success an Agreement is created.

The negotiation process between the Meta-Scheduling Service and the Adapters is implemented based on a proprietary protocol. This is done because the WS-Agreement protocol does neither specify means to change an existing agreement nor to cancel an agreement once it was created. This point is covered again in Section 4. The creation of an agreement is done in a synchronous manner. In our case it is feasible because one important goal in the development of the Meta-Scheduling Service was to make the negotiation process between the subsystems as effective as possible and therefore minimise the overhead of the negotiation as much as possible.

## 4 Requirements for Negotiation of Service Level Agreements

In this subsection, we describe our wish-list with respect to the current draft. A smaller part is under consideration for the current WS-Agreement recommendation while the bigger part will be addressed in subsequent versions of WS-Agreement. A **cancellation mechanism** is required to meet a needs of multi-site allocation use case for Business Grid. This is being integrated into the current specification. With regards to **re-negotiation**, we feel that there is a need to distinguish between two types of re-negotiation: (i) Re-negotiation is done in the initial phase where the two parties negotiate on the Agreement terms and (ii) re-negotiation is needed after an Agreement has reached the *Observed* state.

Here we would like to discuss two issues for the latter re-negotiation type, with the assumption that the current Agreement is in the Observed state during the re-negotiation. There are two major features we think are necessary to make WS-Agreement an even more useful specification for SLAs: a modifiable expiration time and a modifiable list of resources. The first requirement is able to satisfy requests arising from dynamic business conditions, where one party may want to extend or shorten the time that an Agreement will be effective. The second requirement addresses situations - like the one described before in the Business Grid example - where resources will be allocated within a certain range specified in order to meet the SLAs. However, due to increasing resource demand there may be cases in which the Agreement Initiator would like to plan for enhancements to the current system. This could be done in the following two alternative ways: Re-negotiation of the original Agreement or creation of a new Agreement with references to the original Agreement.

The discussion of the items of the wish-list has recently been started in the GRAAP working group. As a result of this discussion several small subgroups (“design-teams”) have already been installed or will be installed soon. These sub-groups focus on

- the further development of WS-Agreement to include e.g. cancellation and re-negotiation,
- a protocol for a multi-step negotiation of agreements as used for example in the VIOLA project,
- provision of other term languages, e.g. including network related terms, and
- terms related to reservation, e.g. start, duration, strategy.

## 5 Future Perspectives

Even before the WS-Agreement specification has become a proposed recommendation of the GGF early adopters have already started using WS-Agreement prototypes in several projects, e.g. Cremona [9]. It has also attracted attention from several large communities, as described above, and we expect a broader uptake once the specification is published as proposed recommendation.

The next major activity to be carried out is the demonstration of interoperability between different implementations of WS-Agreement. In parallel the GRAAP Working Group will start to work on a negotiation protocol based on WS-Agreement, where it is planned to include most of the features described in Section 4. We already started discussion and work on the protocol for the multi-step negotiation between resource provider and resource consumer. Finally, the third activity started recently is focusing on new term languages to support a broader range of usage scenarios for WS-Agreement, e.g. expressing domain-specific service level objectives for network or security.

## 6 Acknowledgements

Some of the work reported in this paper is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #01AK605L. This paper also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST programme under grant #004265. In addition, part of this work had been funded by the Ministry of Economy, Trade, and Industry (METI), Japan with administrative support by Information-Technology Promotion Agency (IPA), Japan. The authors would like to thank Hiro Kishimoto, Andreas Savva, Nobutoshi Sagawa, Shinya Miyakawa for various discussions.

## References

- [1] 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), September 2005. 3 Apr 2006 <<https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecificationDraft.doc/en/26>>.
- [2] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification v1.0. Grid Forum Document GFD.56, Global Grid Forum, November 2005.
- [3] The Business Grid Project. 30 Mar 2006 <<http://www.ipa.go.jp/english/softdev/sixth.html>>.
- [4] EGEE – Enabling Grids for E-sciencE. 3 Apr 2006 <<http://public.eu-egee.org/>>.
- [5] GGF – The Global Grid Forum. 3 Apr 2006 <<http://www.ggf.org>>.
- [6] Grid Resource Allocation Agreement Protocol Working Group. 3 Apr 2006 <<https://forge.gridforum.org/projects/graap-wg/>>.
- [7] The IPsphere FORUM. 3 Apr 2006 <<http://www.ipsphereforum.org/home>>.
- [8] S. Loughran et al. Configuration Description, Deployment, and Lifecycle Management (CDDL) Deployment API. Grid Forum Document GFD.69, Global Grid Forum, March 2006.
- [9] H. Ludwig, A. Dan, and B. Kearney. Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In M. Aiello, M. Aoyama, F. Curbera, and M. Papazoglou, editors, *Proc. of the 2nd International Conference on Service Oriented Computing (ICSOC 2004)*, pages 65–74. ACM Press, 2004.
- [10] A. Streit, O. Wäldrich, Ph. Wieder, and W. Ziegler. On Scheduling in UNICORE – Extending the Web Services Agreement based Resource Management Framework. In *Proc. of Parallel Computing 2005 (ParCo 2005)*, Malaga, Spain, September 13–16, 2005. To appear.
- [11] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN. 29 Mar 2006 <<http://www.viola-testbed.de/>>.
- [12] O. Wäldrich, Ph. Wieder, and W. Ziegler. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, editors, *Proc. of the 2nd Grid Resource Management Workshop (GRMWS'05) in conjunction with PPAM 2005*, volume 3911 of LNCS, pages 782–791. Springer, 2006.
- [13] OASIS Web Services Resource Framework (WSRF) TC. 3 Apr 2006 <<http://www.oasis-open.org/committees/wsrf>>.