

Web Services Agreement-based Resource Negotiation in UNICORE*

M. Riedel, V. Sander, P. Wieder
Central Institute of Applied Mathematics
Research Centre Jülich
D - 52425 Jülich, Germany

J. Shan
Nat. High Performance Computing Center
Univ. of Science and Technology of China
230027 Hefei, China

Abstract

Service Level Agreements provide the foundation to negotiate for a distinct Quality of Service level between the provider and the consumer of a service. Since the Grid community is adopting concepts of Service-Oriented Architectures and Web Services are capturing their space within the Grid landscape, resource management within Grids increasingly evolves towards the management of resources represented as services. To make allowance for this the Global Grid Forum develops the Web Services Agreement specification to support standardised creation and negotiation of guarantees related to services. This paper illustrates the integration of a Web Services Agreement-based resource management framework into the UNICORE Grid system, a development motivated by the system's transition towards a service-oriented Grid and the limitations of the current solution.

Keywords. SLA, SOA, UNICORE, UniGrids, Web Services, WS-Agreement

1 Introduction

At the present time the Grid community seems to have a clear notion of what properties and functions future Grid systems should comprise [12]. Concerning the underlying technology and the architectural model of forthcoming Grids, there is a trend, which propagates the usage of Web Services and the adoption of Service-Oriented Architecture (SOA) concepts and models. The development towards the integration of Grids, Web Services and SOA has been well-founded by the introduction of the Open Grid Services Architecture (OGSA, [10]), an initiative advanced by the respective Global Grid Forum working group.

Currently scientists, researchers and vendors design and develop multiple Grid architectures and systems based on the Grid Service paradigm and with OGSA concepts in mind. One of these systems is UNICORE [5], initially designed as a Uniform Interface to Computing Resources. With the expansion and abstraction of the term resource beyond computing and data, the focus of the UNICORE development changed towards service-orientation. Projects and the community make efforts to enable this transition [14], an objective supported by UNICORE's models and architecture which are designed in a service-oriented way as analyses reveal [17, 16].

The work reported in this paper concentrates on one of the crucial aspects in the current Grid development: resource management. The requirements for resource management services are described by a group of independent experts convened by the European Commission who envisage “flexible, dynamic,

*This work is partially funded by the UniGrids project under EC grant, duration: July 2004 - June 2006.

reconfigurable resources available on demand and to the level required for the application and / or end-user" [12] for the next generation of Grids. Our work integrates appropriate models and Grid Services into the UNICORE framework to provide resource management with properties and Quality of Service capabilities as mentioned above.

Following the introduction the scene is set in Section 2 where we present UNICORE's current resource management model and present the need for enhanced mechanisms to satisfy both end-users and resource providers. Section 3 then introduces the underlying technology of the chosen solution, namely Service Level Agreements, the Web Services Agreement specification and the Web Services Resource Framework (references are provided in the respective section). How these technologies are used to provide the requested resource management capabilities is outlined in Section 4, followed by an analysis of the chosen solution in Section 5. The paper concludes with an outlook on the potential of the WS-Agreement-based resource negotiation in UNICORE, especially with respect to multi-phase negotiation.

2 Resource Management in UNICORE

With reference to the essential resource management properties and capabilities introduced in Section 1, we outline more precisely the corresponding demands made on a state-of-the-art system to manage resources within a Grid:

- **Dynamic information** End-users (or the respective entities acting on their behalf) must operate on up-to-date resource information in order to make a reasonable resource selection.
- **QoS guarantees** End-users and service providers require Quality of Service (QoS) guarantees when participating in the Grid-wide resource sharing.
- **Resource autonomy** Service providers demand autonomous control of the resources they share in a Grid environment. This is achieved by policy-based resource provisioning.
- **Economic models** Economic based resource management [6] will play an important role in Grid Computing and at least the necessary foundations should be integrated into the design of future resource management systems.

These demands define the basis for the design and realisation of advanced UNICORE resource management models, protocols and services.

With regard to the provision and usage of resources UNICORE offers two major models: resource virtualisation and job abstraction [8]. In UNICORE a set of resources with a uniform user mapping and direct access to each other is called a *Virtual Site* (Vsite). As a rule these resources are controlled by a single resource management system. Furthermore, a *UNICORE Site* (Usite) represents a single access point to a set of Vsites (at least one), making it the building block of a Virtual Organization of UNICORE controlled resources. In technical terms a Usite is accessible via a unique address and port generally virtualising the resources of one institution.

The job model distinguishes between abstract and *incarnated* representations of a UNICORE job. The former defines a job description which contains no site-specific information and is processable by every Usite, while the latter job representation comprises everything to actually execute a job on resource level. In-between (arriving at its target Vsite) the abstract job is "translated", a process called *incarnation*. For incarnation as well as for the composition of jobs and workflows resource-specific information is needed. This information is currently maintained per Vsite in a static text file, the *Incarnation Database* (IDB). It contains inter alia information about *capability resources*, such as software distributions, and about *capacity resources*, such as memory or nodes available.

During initialisation a UNICORE server component named *Network Job Supervisor* (NJS, please refer to Section 4) reads the information contained in the IDB into memory and keeps it unchanged during its lifetime. This implies that changes of the resource situation during runtime are not reflected in the resource information of the UNICORE system. When connected to a Usite, the client-side application submits a `GetResourceDescription` request to the NJS and the corresponding handler `DoGetResources` within the NJS reads the information from memory and returns it back to the client-side application. After obtaining the resource information, end-users or the respective client application can make a match between their jobs' requirements and the available resources. Finally, the end-users will submit their jobs, which contain the specified resource requirements, to the Usite. Although this model supports end-users to perform their job submission, it only provides a static resource view of Vsites and can thus not meet the requirement of providing dynamic resource information. In addition, the current UNICORE system does neither allow resource users and providers to negotiate on QoS guarantees nor does it include the means to manage resources based on economic models. And even though UNICORE fulfils the "Resource autonomy" requirement with respect to the autonomous control, no framework exists e.g. to enforce a certain policy on a single user. This gap analysis motivated the work described here and led to the design and implementation of a WS-Agreement based resource negotiation model to meet the requirements mentioned above.

3 Service Level Agreements for Resource Negotiation

As noted in the previous section the management of QoS guarantees is a functional requirement future Grid systems have to satisfy. In a service-oriented context QoS management refers to the negotiation, enforcement and monitoring of contracts between a service requestor and a service provider. The objective of such contract negotiations is to agree upon specific capabilities provided by a service, thus to define an implicit level of service. This is of major importance during the selection of a service provider since its actual capabilities might depend on the resource situation at the requested time of service. By modeling the implicit level of service explicitly as a Service Level Agreement (SLA) we formalize the relation between service requestor and provider. This leads to the problem of formally defining agreement terms and a standard way of negotiating them.

While several SLA languages and protocols, such as the Web Service Level Agreement (WSLA) language [13] or the Business Process Execution Language (BPEL, [1]), can be used to support resource negotiation in Grids, we have chosen the development of the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG) of the Global Grid Forum. Their proposed WS-Agreement specification [2] defines a language and a protocol for advertising the capabilities of service providers and creating SLAs based on agreement templates and offers. The well defined agreement structure in combination with the straightforward negotiation process based on templates and offers is the reason why we preferred WS-Agreement to WSLA or BPEL. The agreement creation process starts with a

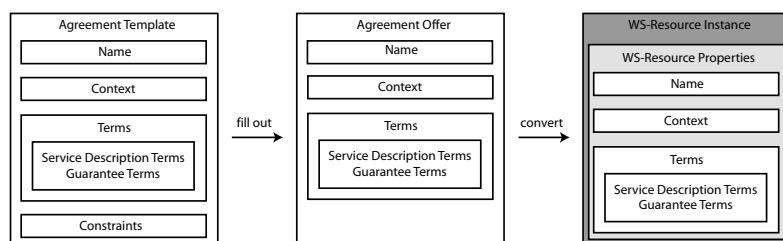


Figure 1. Transforming a WS-Agreement into WS-ResourceProperties

pre-defined agreement template specifying customizable aspects of an agreement, as there are *Name*, *Context*, *Terms* and *Constraints* (see Fig. 1). While the *Constraints* enforce rules that must be observed

creating an agreement, the Context contains information about the participants in the agreement. The Terms section is split in *Service Description Terms* (SDT) and *Guarantee Terms* (GT). While SDTs, also called functional terms, describe the essential characteristics of a service, the GTs define assurances on service quality associated with the service described by the SDTs.

The service provider advertises the types of agreement offers it is willing to accept by means of such agreement templates. A service requestor then fills in a template and sends it to the provider, a process resulting in a concrete agreement offer, which is accepted or rejected dependent on the actual resource situation of the service provider. Hence, this simple template and offer mechanism provides a dynamic resource negotiation framework to be used in Grid systems and especially in UNICORE.

If an offer is accepted the service provider creates a stateful *WS-Resource* [9] wherein the state represents the structure of the agreement. The agreement offer is then converted into *WS-ResourceProperties* [11] as Fig. 1 shows. Thus, the WS-Agreement specification relies on OASIS' Web Services Resource Framework (WSRF, [7]) that includes the WS-ResourceProperties specification. WS-ResourceProperties defines the representation of properties inside a WS-Resource as well as a standard set of message exchange patterns that allow a service requestor to query resource properties. WS-Agreement uses this specification to query agreement properties, especially SLA terms, at service runtime.

4 Dynamic Resource Management Architecture

Based on the definition of the resource management requirements and the related UNICORE gap analysis in Section 2, we now evolve the integration of the WS-Agreement framework as described in Section 3 into the UNICORE architecture.

4.1 Basic UNICORE architecture

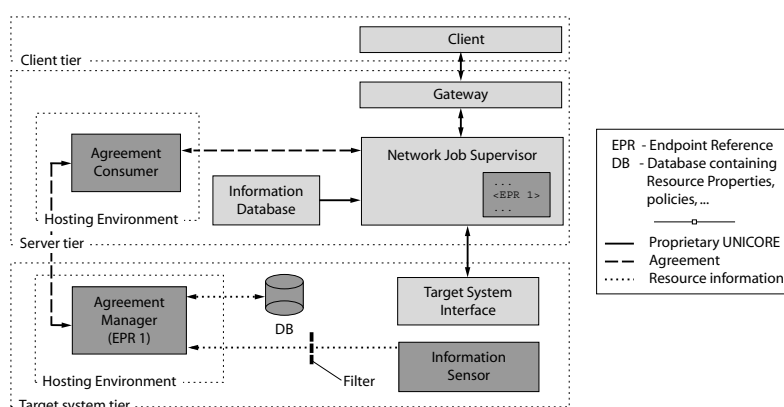


Figure 2. UNICORE components (light grey) and architecture enhancements (dark grey)

As shown in Fig. 2, UNICORE introduces a three tier Grid architecture consisting of client, server and target system tier, an implementation of which is realized entirely in Java (except for the *Target System Interface* (TSI) which is implemented in Perl). This paragraph introduces the basic concepts and components (those in light grey) relevant for resource management, for an extensive insight please refer to [8]. Section 4.2 then pictures the enhancements implemented to realise the dynamic resource management architecture, while Section 4.3 depicts selected implementation details. The client tier is either represented by an application making use of the UNICORE client API or by the *UNICORE Client* which provides a graphical user interface to exploit the entire functionality offered by the server tier. This is achieved by sending and receiving abstract representations of UNICORE jobs as introduced in Section 2. In addition to resource information this job instantiation contains platform and site neutral

descriptions of computational and data related tasks as well as work flow specifications along with user and security information.

The *Gateway* is the secure entry point to a *U-site*, which authenticates user requests and transfers them to the *Network Job Supervisor*. The NJS translates the abstract job into a target system specific one, making use of the Incarnation Database. Furthermore the NJS contains a work flow engine which, among other tasks, performs pre- and post-staging of computational data and authorises the user. The Gateway and NJS execute typically on dedicated secure systems behind a firewall.

UNICORE's communication endpoint is the Target System Interface (TSI), a stateless daemon executing on the target system. It interfaces with the local resource manager represented either by a batch system like LoadLeveler, an operating system like Linux (using fork) or an entity capable of inter-operating with other Grid systems like Globus [18]. In a typical setup one TSI and one NJS form a *V-site* (see Section 2).

4.2 Architecture Extensions

Extending the three tier Grid architecture of UNICORE, Fig. 2 depicts several components in dark grey that have been added to the basic architecture. These extensions realise a UNICORE-specific implementation of the WS-Agreement framework, which itself defines a two-layered service model [2], namely *service layer* and *agreement layer*.

The service layer is integrated inside the target system tier, represented by an *Information Sensor* (IS) which exploits dynamic resource-specific information and therefore fulfils the first demand specified in Section 2. To enforce user-specific resource management and support the provider's autonomy we implemented a filter that makes use of policies stored in a database (DB). This filter allows e.g. to restrain information delivered to certain users or restrict access to certain parts of the resource.

The agreement layer consists of two components, the *Agreement Manager* on the target system tier and the *Agreement Consumer* on the server tier. These components enable an UNICORE Client to submit a resource request and negotiate the terms of resource usage. This is accomplished by forwarding any resource information request from the UNICORE Client via the NJS directly to the Agreement Consumer, which in turn enquires for resource information provided by the Agreement Manager. The Agreement Manager responds by using the service layer to get the actual resource information from the IS. In particular, the Agreement Manager generates a WS-Agreement compliant template containing the resource information which is transferred back to the Agreement Consumer. The requested template contains the current, filtered information about a UNICORE resource represented as SDTs. For example in the case of a computing resource the template comprises, among others SDTs, representing information like nodes, processors or memory. The subsequent step in this negotiation process requires the Agreement Consumer to fill in the template to make a concrete agreement offer to the Agreement Manager which then is in charge of creating and managing the agreement. The completion of the template is supported by the UNICORE Client. This agreement offer constitutes a concrete resource request that leads to the creation of an UNICORE resource agreement instance if the Agreement Manager accepts this offer. This UNICORE resource agreement instance provides access to dynamic resource information through the UNICORE Client.

4.3 Prototype Implementation

The implementation of the afore outlined resource negotiation architecture mainly complies two tasks: the modification of the original UNICORE software and the implementation of the current WS-Agreement specification. Since the modifications of the UNICORE software imply primarily protocol extensions, we will concentrate on the integration of a WSRF-compliant hosting environment and WS-Agreement-related Grid Services into UNICORE as shown in Fig. 3.

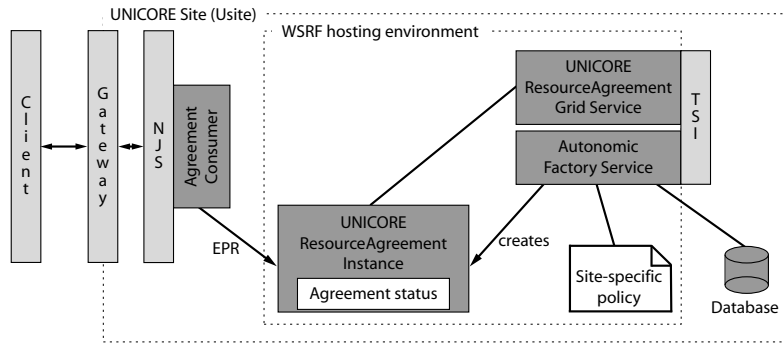


Figure 3. Services implementation details

WSRF uses the factory pattern [7] to create WS-Resources that are capable of storing the resource's state. We realise this pattern as an *Autonomic Factory Service* [15] that implements the `AgreementFactory` portType [2] and provides the `createAgreement` and `GetResourceProperty` operations to the Agreement Consumer. While `GetResourceProperty` provides access to the WS-ResourceProperties which are dynamically updated by the IS, the `createAgreement` operation generates a *UNICORE ResourceAgreement Instance* and an *Endpoint Reference* (EPR [4]) that can be used to access this WS-Resource through the *UNICORE ResourceAgreement Grid Service*. In consequence the agreement state is represented by the UNICORE ResourceAgreement Instance and manipulated via the UNICORE ResourceAgreement Grid Service that implements the Agreement portType [2]. This portType defines a `Terminate`, `GetResourceProperty` and other domain-specific operations, like for instance `QueryResourceProperties` [11]. Furthermore, the portType exposes WS-ResourceProperties, namely Name, Context, and Terms which are converted from the Agreement offer (see Section 3) into the WS-ResourceProperties of the UNICORE ResourceAgreement Instance. These architecture enhancements lay the foundations for dynamic resource management in UNICORE and, among other functionality, enable user-specific resource management and support the provider's autonomy through site-specific policies managed by the Autonomic Factory Service, as the following sections reveal.

5 Architecture Evaluation

As determined in Section 2 the purpose of the work reported here is to enhance UNICORE's resource management capabilities with functions to advertise dynamic resource information, enable the negotiation of QoS guarantees, enable economic resource management and retain resource autonomy. These functions and their impact on UNICORE will be briefly described here.

Using up-to-date resource information UNICORE users are now in the position to make a suitable resource selection based on the actual resource situation on the target system. Although this still implies jobs to be scheduled manually by the user, the transition from a static to a dynamic information service lays the foundation for more advanced scheduling architectures as researched in projects like VIOLA (Vertically Integrated Optical Testbed for Large Applications).

Moreover, our work provides the means for users or applications to demand a certain service level. Currently this service level is negotiated within one step, but once the respective WS-Agreement Negotiation specification [3] is available, multi-step negotiation will be integrated into the UNICORE framework.

Apart from carrying out an arbitrary number of steps to agree upon the modalities of resource usage, economic resource management requires the possibility to reserve resources in advance. This capability can be easily realised with the described architecture by including additional service description terms which represent the reservation and are defined inside the agreement.

The introduction of site-specific policies overcomes the limitations of the current situation in which all users have the same rights and therefore strengthens the autonomy of resource providers. As an extra benefit, since all service description terms of an agreement are exposed as WS-Resource properties, a user or application can monitor agreement-compliance at the run-time of the service.

6 Conclusion and Future Work

An evaluation of the UNICORE framework showed that its resource management capacity lacks certain functionality with respect to the requirements users and experts have. To fulfil these requirements we designed and implemented a dynamic resource management architecture to extend the UNICORE system. This architecture is based on the WS-Agreement specification and the WS Resource Framework and enables users to control and to manage dynamic resource allocation through agreements. Furthermore the proposed solution is extensible and therefore allows the integration of various different resources and their respective QoS parameters.

The previous section shows not only that our solution improves usage and administration of the UNICORE system, but it also provides the basis to realise functions like advance reservation, multi-step negotiation, related agreements, automatic scheduling and support for dynamic workflows. Hence we will exploit the potential of this work and develop additional services and components to make, as stated in the introduction, “*flexible, dynamic, reconfigurable resources available on demand*”.

References

- [1] T. Andrews et al. Business Process Execution Language for Web Services, 2003. Version 1.1.
- [2] A. Andrieux et al. Web Services Agreement Specification, 2004. Version 1.1, draft 20.
- [3] A. Andrieux et al. Web Services Agreement Negotiation Specification, 2005.
- [4] D. Box, F. Curbera, et al. Web Services Addressing (WS-Addressing), 2004.
- [5] D. Breuer, D. Erwin, D. Mallmann, R. Menday, M. Romberg, V. Sander, B. Schuller, and P. Wieder. Scientific Computing with UNICORE. In D. Wolf, G. Münster, and M. Kremer, editors, *Proc. of the NIC Symposium 2004*, volume 20, pages 429–440. John von Neumann Institute for Computing, 2004.
- [6] R. Buyya and H. Stockinger. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. In *SPIE International Symposium on The Convergence of Information Technologies and Communications (ITCom 2001)*, 2001.
- [7] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. The Web Services Resource Framework, 2004. Version 1.0.
- [8] D. Erwin, editor. *UNICORE Plus Final Report – Uniform Interface to Computing Resources*. UNICORE Forum e.V., 2003. ISBN 3-00-011592-7.
- [9] I. Foster et al. Modeling Stateful Resources with Web Services, 2004.
- [10] I. Foster, C. Kesselmann, J. M. Nick, and S. Tuecke. The Physiology of the Grid. In F. Berman, G. C. Fox, and A. J. G. Hey, editors, *Grid Computing*, pages 217–249. John Wiley & Sons Ltd, 2003.
- [11] S. Graham, J. Treadwell, et al. Web Services Resource Properties, 2004.
- [12] K. Jeffery et al. Next Generation Grids 2 – Requirements and Options for European Grids Research 2005-2010 and Beyond, July 2004.
- [13] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement Language Specification, 2003. Version 1.0.
- [14] R. Menday and P. Wieder. GRIP: The Evolution of UNICORE towards a Service-Oriented Grid. In *Proc. of the 3rd Cracow Grid Workshop (CGW’03)*, Oct. 27–29 2003.
- [15] M. Riedel. Prospects and Realization of Flexible Service Offers in a Grid Environment. Technical Report JUEL-4154, Research Centre Jülich, 2004.
- [16] D. Snelling. UNICORE and the Open Grid Services Architecture. In F. Berman, G. C. Fox, and A. J. G. Hey, editors, *Grid Computing*, pages 701–712. John Wiley & Sons Ltd, 2003.
- [17] D. Snelling, S. van den Berghe, G. von Laszweski, P. Wieder, D. Breuer, J. MacLaren, D. Nicole, and H.-C. Hoppe. A UNICORE Globus Interoperability Layer. *Computing and Informatics*, 21:399–411, 2002.
- [18] P. Wieder and M. Rambadt. UNICORE – Globus: Interoperability of Grid Infrastructures. In *Proc. of the Cray User Group Summit 2002*. Cray User Group, 2002.