

# GRIP: The Evolution of UNICORE towards a Service-Oriented Grid

R. Menday, Ph. Wieder

Central Institute for Applied Mathematics, Research Centre Jülich, Jülich, Germany

*email:* [r.menday,ph.wieder]@fz-juelich.de

*phone:* (+49 2461) 612760, *fax:* (+49 2461) 616656

## Abstract

The UNICORE Grid system implements a vertically integrated Grid architecture providing seamless access to resources within different organizations. The software is developed and deployed by companies, research - and computing centres and projects throughout Europe. The UNICORE Forum [1] promotes the UNICORE software and interfaces, and makes it available for download. In this paper we instance UNICORE for the evolution of a Grid system towards a service-oriented Grid, primarily focussing on architectural concepts and models. Based on the current architecture and the enhancements provided by the project GRIP [2], we depict first steps already taken to integrate Web - and Grid Services into UNICORE. This includes provision of OGSI-compliant portTypes [3] parallel to the proprietary interfaces as well as the design of XML-based protocols. Furthermore we present the roadmap defined by GRIP to achieve a consistent development towards an OGSA [4] implementation.

## 1 Motivation and challenges

Observing the development of the World Wide Web, the Grid-to-be should comprise of standards-compliant services from multiple providers. Standards compliance is the crucial factor to master the following challenges:

- **Seamlessness** As with the browser and the WWW, users should be able to access the Grid seamlessly, securely and easily using their tool of choice.
- **Autonomy** Resource owners may choose which resources to release onto the Grid, and making use of their preferred hosting environment.
- **Independence** Developers of Grid middleware decide on platform and programming environment to implement services.

Compared with the seemingly unrestrained adoption of the World Wide Web, the Grid has not yet ‘taken-off’. Currently we observe little evidence of true interoperability between Grid systems<sup>1</sup>, and consequently a limited applicability

---

<sup>1</sup>An exception is the Grid Interoperability Project (GRIP, see [5]), which is addressing interoperability between UNICORE and Globus [6].

and number of users. Non-standard, monolithic, hard-to-install and hard-to-configure architectures lack the adaptability, scalability and portability essential for the Grid to realise its full potential of *available everywhere*, internet scale computing [7].

UNICORE is a fully-fledged and powerful Grid system. That it is sometimes categorised as just a user-friendly graphical interface to Grid resources shows some misunderstanding. Moreover in some articles UNICORE is classified as a kind of portal (see e.g. [8]), which is perhaps due to the high visibility of the user-interface. Indeed many perceive the UNICORE client to be intrinsically inseparable from the UNICORE server components, giving the impression of a closed system. It is a complaint sometimes made by other Grid practitioners who would like access resources federated by UNICORE, but not necessarily through the UNICORE client. Whilst the UNICORE client API makes this possible, the interfaces and protocols between the various components are at best published but not standardised. This is how UNICORE has come to be described as being vertically integrated.

Whilst some characteristics described here are those of a tightly-coupled architecture and may seem to conflict with service-oriented philosophies, we believe that many of the underlying principles, abstractions and models of UNICORE are congruent with those of the Open Grid Services Architecture (OGSA). The benefits a service-oriented approach may bring to UNICORE are described in this paper along with some of the issues, ideas and proposals in order to move UNICORE towards a service-oriented architecture.

## 2 The UNICORE Grid system

As shown in Figure 1 UNICORE introduces a three tier Grid architecture consisting of user, server and target system tier, an implementation of which is realized entirely in Java<sup>2</sup>. This paper introduces the basic concepts and components, for an extensive insight please refer to [9].

The user tier is represented by the UNICORE Client<sup>3</sup> which provides a graphical user interface to exploit the entire functionality offered by the server tier. This is achieved by sending and receiving Abstract Job Objects (AJO) via the UNICORE Protocol Layer (UPL). The AJO is the realisation of the job model and central to UNICORE's philosophy of abstraction and seamlessness. It contains platform and site neutral descriptions of computational and data related tasks, resource information and workflow specifications along with user and security information. AJOs are built within the user tier and sent formatted as serialized and signed Java objects to the Gateway.

---

<sup>2</sup>A Perl implementation of the Target System Interface (TSI) also exists to cover mainly HPC systems with no Java VM.

<sup>3</sup>In addition a client API exists which is used by several projects to integrate user tier functions.

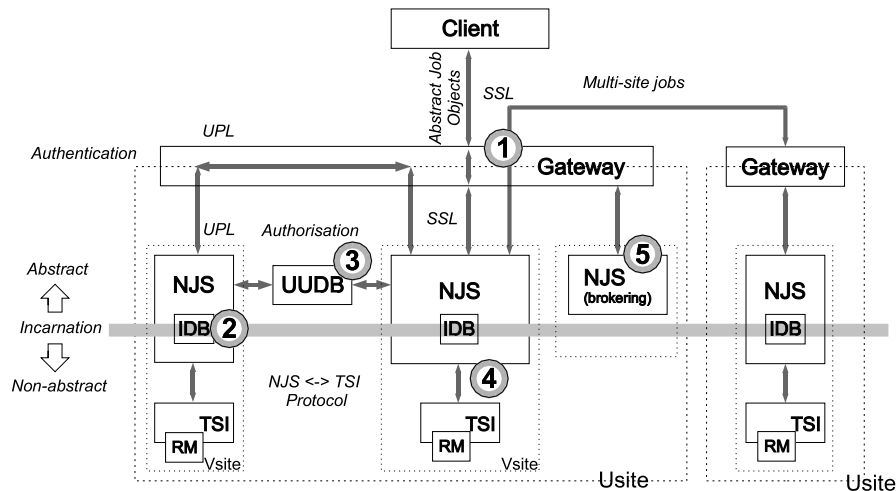


Fig. 1: UNICORE architecture, including Grid Service interfaces developed in GRIP: (1) UPL, (2) IDB, (3) UUDB, (4) NJS – TSI, and (5) Broker.

The Gateway is the secure entry point to a UNICORE site, a *Usite*, which authenticates user requests and transfers them to the Network Job Supervisor (NJS). The NJS translates the abstract job into a target system specific one, a process called *Incarnation*, making use of the Incarnation Database (IDB). Furthermore the NJS contains a workflow engine which, among other tasks, performs pre- and post-staging of computational data and authorises the user by contacting the UNICORE User Database (UUDB). 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, a batch system emulation on top of Linux (for example) or an entity capable of providing access to Grid resources like Globus.

Permanent X.509 certificates are used to establish SSL connections between UNICORE components<sup>4</sup> and to sign Abstract Job Objects (see [10] for a full analysis of UNICORE's security model).

### 3 Web and Grid Services

Web Services offer a relatively simple, XML-based framework for doing distributed computing. HTTP is always supported as a transport protocol, but other transport mechanisms can be used or implemented too. These simple

<sup>4</sup>E.g. the above mentioned UPL is layered on top of SSL.

foundations show Web Services can be built on an infrastructure similar to the World Wide Web - an indicator of future success.

The basic specifications are the Web Services Description Language ([11], WSDL) and SOAP [12], for the description and invocation of Web Services. Other specifications in the areas of messaging, security and transactions build on top of the base specifications, and are intended to be used in a modular fashion. We are particularly interested in leveraging Web Service security (WS-Security [13], Security Assertion Markup Language (SAML) [14]), messaging (WS-Routing [15], WS-Addressing [16]), discovery (WS-Inspection [17]) and possibly Web Service workflow descriptions (e.g. BPEL4WS [18]).

## Relevant Grid Service standardisation efforts

The Global Grid Forum is the “community-initiated forum ... working on distributed computing, or ‘grid’ technologies” [19]. Although it does not aim at defining standards but ‘best-practices’, its recommendations and contributions have a major impact on the development of the Grid or may be considered as de-facto standards as e.g. [3].

Of primary interest to our work is the Open Grid Services Architecture WG [20]<sup>5</sup> which documents the requirements, functionality, priorities and interrelationships for OGSA services. Some of the groups providing the foundations to realize an OGSA-compliant architecture are listed below. Their possible impact on UNICORE is raised in Section 4 and Section 5:

- **CMM-WG** Defines a Common Management Model and a set of OGSI portTypes for the standardised management of resources and services.
- **GRAAP-WG** The WS-Agreement specification defines agreement negotiation OGSI portTypes for the usage of services.
- **OGSA-SEC-WG** Builds a basis for a Grid Service security framework.
- **GridIR-WG** Specifies an information retrieval system on the OGSA Grid, covering among other things collection management, indexing/searching and query processing.
- **JSDL-WG** Specifies an abstract Job Submission Definition Language and the corresponding XML schema. JSDL will provide the terms to be used with the WS-Agreement specification.

## 4 UNICORE’s architectural options

At approximately the end of the first year of the GRIP project <sup>6</sup>, it pursued a proactive initiative to move UNICORE towards a service-oriented Grid, partly by adopting the Open Grid Service Infrastructure standard recommendation,

---

<sup>5</sup>All GGF groups’ web sites are hosted here. We do not refer to each site individually.

<sup>6</sup>The Grid Interoperability Project (GRIP) is a two years project which started January 1, 2002.

partly by driving GGF activities. In this section, we report on the progress of this work, and take a look in the longer term. In the next section we sketch new opportunities, and the direction where UNICORE could be taken.

## Grid Services developed during the GRIP project

The following is a list of (both designed and implemented) portTypes for UNICORE (see Figure 1). These portTypes offer Web - or Grid Service interfaces to the main building blocks while proprietary interfaces will exist in parallel until the new ones provide the full functionality (and as long as users request compatibility):

- (1) **UPL portType** This interface lets Grid Service enabled clients send Abstract Job Objects (see Section 2) to a UNICORE server via SOAP. Divers implementations hosted in systems such as GLUE [21] or GT3 [6] have been realized.
- (2) **IDB portType** To expose the content of the Incarnation Database a Grid Service is needed which advertises the resource information to the NJS or the broker (or any information sink capable of processing them). To realize this the IDB portType offers IDB Service Data to a subscriber by implementing the OGSi NotificationSource portType <sup>7</sup>.
- (3) **UUDB portType** A UUDB service provides authorization information for the system its corresponding TSI is executing on. Therefore this interface is exposed internally only at the moment, its potential expansion towards a community service is discussed in Section 5.
- (4) **NJS-TSI portType** The interface between NJS and TSI is used to submit the incarnation of a UNICORE job to the target system exposing Grid Service mechanisms. In that case the protocol will also rely on Web Service techniques, defining (proprietary) XML messages to be consumed by the TSI Grid Service.
- (5) **Broker portType** First within the EUROGRID [22] and then the GRIP projects, work has been undertaken to design and build a brokering component for UNICORE. The continuing work in GRIP extends the broker for cross-Grid brokering over heterogeneous resources (currently UNICORE and Globus resources). The intention is to interface the broker as a Grid Service.

## The GRIP roadmap

To guarantee that the development which has been initialized through GRIP will continue after the end of the project, a roadmap has been defined not only covering the interfaces explained above, but extrapolating the transition towards a higher level of interoperability. In order to support virtual organisation structures, OGSi-compliance is only the first step. To gain what GRIP envisions

---

<sup>7</sup>Please refer to [3], Section 11, for details.

standard languages and protocols are as crucial as e.g. a common understanding of terms.

Concerning the GRIP roadmap and the evolution of UNICORE the number of architectural issues to deal with is beyond the scope of this document, but to pick up the paradigms brought up in Section 1 some directions may be discussed. For the user to seamlessly access the Grid, her tool of choice has to specify UNICORE resource requests in a standardised format, for example, using GGF's Job Submission Description Language (see JSDL-WG, Section 3), or at least provide mechanisms to map them to a standard representation, e.g. an Ontology [23]. To retain site autonomy the resource provider needs tools to specify the policies to grant or restrict access to resources, for which an enhanced UADB Grid Service will be the right instrument. Finally, to honour the independence from a distinct programming environment the AJO will be defined in XML not forcing developers to use serialized Java objects.

## 5 Externalising the interfaces - looking to the future

In our opinion UNICORE provides an excellent basis for building a pervasive and service-oriented system where any functionality can be obtained through accessing services (including services comprised of multiple aggregated services). We envisage a *universe of services*, loosely coupled [24], self-describing, standards-compliant, and where each Grid node hosts one or more services. Every node listens for service requests, and when it receives one, continues to interact with the requestor to fulfill the service requirements requested.

### Flattening the hierarchy

In practice, it is observed that UNICORE is accepted more readily by systems administrators, due to its less obtrusive deployment. This firewall friendly aspect of UNICORE will be exploited in any revision of the architecture.

The Gateway component implements the UPL, which is a simple and robust request-response protocol which allows a client to consign an AJO, and then retrieve the status and outcome of its execution. Also, via UPL a client can retrieve a list of Vsites to which a particular Gateway provides access. Aside from responding to UPL requests, the Gateway essentially authenticates the user and passes the message through to the primary NJS indicated in the AJO.

Thus we can view the UNICORE Gateway as a lightweight filtering component, in effect, as SOAP intermediary (or indeed a SOAP gateway). Assuming that message-level security is used, the authentication function of the Gateway is performed by examining the relevant WS-Security element of the SOAP header. Also routing information contained in the header could be used by the Gateway to route the message to the desired component. This provides a global addressing for services which are located behind intermediaries, and has the effect of

flattening the hierarchy which is presently imposed by UNICORE. Finally, the list of available Vsites could be published using WS-Inspection interfaces, or via the Service Data mechanism of OGSi [3].

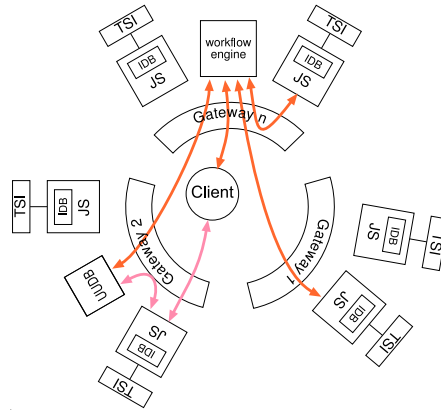


Fig. 2: Another view of the UNICORE architecture

### Decomposing UNICORE's functionality

A direct consequence of evolving a looser coupling between the components of UNICORE is that it encourages others to exploit and utilise the services and resources federated by UNICORE. For example, the exposure of UNICORE resources, which is in effect the ability to execute an 'atomic' UNICORE job). This requirement points to the splitting of the NJS component: one part as a workflow coordinator, and the second as a job executor. This is shown in Figure (2) - the NJS becomes the 'workflow engine', and the 'JS' (Job Service).

Although the principle of representing jobs in an abstract form is very good, one must question tying the representation of the AJO to the serialisation mechanism of Java. Whilst initially it is most likely that the 'workflow engine' service would take as input serialised AJO's, there would be definite advantages in the future of adopting a standardised workflow description, either coming from the Grid or Web Services communities <sup>8</sup>.

### Security

As discussed in section 4, we would like to develop the UUDB component into more of a community service. It would exist as a service to the whole Usite

<sup>8</sup>In the future it is likely that the distinction between the two will fade.

or (potentially) a collection of Usites (see Figure 2). One proposal is that this would operate by issuing signed assertions according to the security credentials presented to it. A potential specification which would be useful here is SAML, [14]. The current UNICORE architecture could directly use this system in a manner similar to how it does so currently - the signed response assertion contains the xlogin as an attribute. Additionally, the signed assertion could be used as a security token in its own right, and included in a request made of another resource by a component (for example, the workflow engine) on behalf of a user, and hence facilitate a partial delegation mechanism for UNICORE.

The implementation of a community security service and the logical split in the functionality of the NJS, would have the implication of ‘opening up’ the services offered by UNICORE. This development would make it much easier for UNICORE to be used in a non-prescribed manner - for example, from a non-UNICORE component.

## 6 Concluding remarks

Through the GRIP project we have actively participated in a number of standardisation activities, and particularly with the advent of OGSA, we view interoperability in a broader sense not limited to interoperation between UNICORE and Globus.

From our point of view, realising the World Wide Grid implies an evolution from integrated to service-oriented architectures. This paper describes a current state-of-the-art development aiming to realise this objective, but it also makes assumptions and leaves other issues open. It is beyond question that the Open Grid Services Architectures is yet evolving and we consider this paper as a contribution to that discussion.

### Acknowledgements.

The work described here has been funded in part by the Information Society Technologies (IST) Programme of the European Commission under Contract IST-2001-32257. UNICORE Plus was funded in part by BMBF grant 01 IR 001 A-D [9]. We would like to thank all GRIP partners and subcontractors, namely (in alphabetical order) Argonne National Laboratory, Deutscher Wetterdienst, Forschungszentrum Jülich, Fujitsu Laboratory Europe, Intel, University of Manchester, University of Southampton, and Warsaw University.

## References

1. “The UNICORE Forum”, Web Page. Online: <http://www.unicore.org/>.
2. D. Snelling, S. van den Berghe, G. von Laszweski, Ph. Wieder, D. Breuer, J. MacLaren, D. Nicole, and H.-Ch. Hoppe, “A UNICORE Globus Interoperability Layer”, *Computing and Informatics*, vol. 21, 2002, pp. 399-411.

3. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, eds., "The Open Grid Services Infrastructure (OGSI) – Version 1.0", Grid Forum Document GFD-R-P.15, Global Grid Forum, 2003.
4. I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
5. "The Grid Interoperability Project", Web Page. Online: <http://www.grid-interoperability.org/>.
6. "The Globus Alliance", Web Page. Online: <http://www.globus.org/>.
7. I. Foster, "Internet Computing and the Emerging Grid", Nature Web Matters, 2000. Online: <http://www.nature.com/nature/webmatters/grid/grid.html>.
8. J. Novotny, "The Grid Portal Development Kit", *Concurrency and Computation: Practice and Experience. Special Issue: Grid Computing Environments*, vol. 14, issue 13-15, Nov./Dec. 2002, pp. 1129-1144.
9. D. Erwin, ed., "UNICORE Plus Final Report - Uniform Interface to Computing Resources", The UNICORE Forum e.V., ISBN 3-00-011592-7, 2003. Online: <http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf>.
10. T. Goss-Walter, R. Letz, Th. Kentemich, H.-Ch. Hoppe, and Ph. Wieder, "An Analysis of the UNICORE Security Model", Grid Forum Document GFD-I.18, Global Grid Forum, 2003.
11. "Web Services Description Working Group", Web Page. Online: <http://www.w3.org/2002/ws/desc/>.
12. "XML Protocol Working Group", Web Page. Online: <http://www.w3.org/2000/xp/Group/>.
13. "OASIS Web Services Security Technical Committee", Web Page. Online: <http://www.oasis-open.org/committees/wss/>.
14. "OASIS Security Services Technical Committee", Web Page. Online: <http://www.oasis-open.org/committees/security/>.
15. H. F. Nielsen, and S. Thatte, "Web Services Routing Protocol (WS-Routing)". Online: <http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-routing.asp>.
16. D. Box and F. Cubera, eds., "Specification: Web Services Addressing (WS-Addressing)". Online: <http://www-106.ibm.com/developerworks/webservices/library/ws-add/>.
17. K. Ballinger et al., "Specification: Web Services Inspection Language (WS-Inspection) 1.0". Online: <http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html>.
18. S. Thatte, ed., "Specification: Business Process Execution Language for Web Services Version 1.1". Online: <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>.
19. "The Global Grid Forum", Web Page. Online: <http://www.ggf.org>.
20. "GridForge", Web Page. Online: <http://forge.gridforum.org/>.
21. "GLUE", Web Page. Online: <http://www.themindelectric.com/glue/>.
22. "The EUROGRID Project", Web Page. Online: <http://www.eurogrid.org/>.
23. M. Klein, J. Broekstra, D. Fensel, F. van Harmelen, and I. Horrocks, "Ontologies and Schema Languages on the Web", in *Spinning the Semantic Web*, D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, eds., MIT Press, 2003.
24. D. Orchard, "Achieving Loose Coupling", WebServices.org, 2003. Online: <http://www.webservices.org/index.php/article/articleview/1246/1/24/>.