

# Standardization Processes of the UNICORE Grid System

Morris Riedel\* and Daniel Mallmann†

**Abstract.** *The UNICORE Grid system has been developed since the late 1990s to support distributed computing applications and emerging Grid infrastructures. Over the years, UNICORE has evolved to a full-grown and well-tested Grid middleware system, which today is used in daily production at many supercomputing centers worldwide. Also, the UNICORE technology serves as a solid basis in many European and International research projects. In this paper, we present issues surrounding the integration of standards into the UNICORE Grid system. We summarize here the principal characteristics of the latest Web services-based Unicore/GS release, which provides significant enhancements in the areas of interoperability, standards compliance and functionality.*

## 1. Introduction

There are many issues surrounding the integration of emerging standards into Grid middleware such as the the UNICORE Grid system [35], Globus Toolkit 4 (GT4) [26] or the gLite [7]. Standards are emerging which cover a spectrum of functionality ranging from the standardised access of individual resources such as supercomputers or mass storage servers, to the interoperability between different Grid middleware systems. Appropriate standards bodies are the Global Grid Forum (GGF) [8], W3C [?] and the Organization for the Advancement of Structured Information Standards (OASIS) [14] in terms of proposed recommendations or specifications. In particular, GGF standards going through several public review periods before their so-called Grid Final Documents (GFD) reach the proposed recommendation status and can become later a full recommendation by the widespread implementation of the documents proposed specification (e.g. the GridFTP specification [17]). OASIS standards, on the other hand, are approved within an OASIS Committee, submitted for public review, and implemented by at least three organizations. In addition, some Grid components uses existing or proposed standards of the Internet Engineering Taskforce (IETF) [11]. For instance the Internet X.509 Public Key Infrastructure, and the proxy certificate extension as required by GSI [?]. We refer to emerging standards when working groups from standardization organizations and forums have produced substantial work that is suitable for the standardization process, the public review periods, or have already reached proposed recommendation status for their documents. A well-known set of emerging GGF standards are related to a service-oriented framework for Grid computing. The Open Grid Services Architecture (OGSA) emphasises the usage of so-called Grid services [29]. Even if the status of the OGSA GGF document is informational without providing an explicit specification

---

\*Distributed Systems and Grid Computing Division, Central Institute of Applied Mathematics, Research Centre Juelich, D-52425 Juelich, NRW, Germany, email: m.riedel@fz-juelich.de

†Distributed Systems and Grid Computing Division, Central Institute of Applied Mathematics, Research Centre Juelich, D-52425 Juelich, NRW, Germany, email: d.mallmann@fz-juelich.de

in terms of a real standard, the OGSA concepts have reached a wide acceptance in the Grid community. Previously, the Open Grid Services Infrastructure (OGSI) [36] examined foundation concepts for implementing stateful Web services. More recently, specifications from the Web services community have become increasingly important, leading to the WS Resource Framework (WS-RF) [14], currently being standardized by OASIS. WS-RF refactors and evolves OGSI to exploit new emerging Web services standards, for instance the W3C WS-Addressing specification, and to respond to early implementation and application experiences [28].

The rapid definition and adoption of OGSA and WS-RF allow the UNICORE development community to extend the concepts of the production UNICORE Grid system through the usage of Web services technologies. In particular, the European UniGrids project [1] focusses on the development of a UNICORE system that is compliant with emerging Grid standards, named as Unicore/GS. Unicore/GS provides standardized access to Grid resources and is interoperable with other Grid middleware by the usage of atomic services. This interface is defined by the UniGrids Atomic Services (UAS) set of WSDL. We believe that these that can become, in principle, a powerful addition to the OGSA - Basic Execution Services (BES) GGF specification. In addition, the realization of a wide variety of Unicore/GS clients is necessary to provide users with the capability to submit and monitor Grid jobs. Therefore, UniGrids also develops a Grid Programming Environment (GPE) [10] that provides simple application clients, portal client solutions and a thin client to run on mobile devices, all accessing services that are hosted by modern Grid middleware such as Globus or Unicore/GS, in particular the UniGrids.

We summarize here the principal characteristics of the latest Web services-based Unicore/GS - the next successful evolution of the UNICORE software. The remainder of this paper is structured as follows. In Section 2, we provide an overview of the production UNICORE Grid system and give examples of early Web services-based prototypes and their underlying technology in Section 3. The integration of emerging standards into the latest release of Unicore/GS is described in Section 4 while this paper ends with some concluding remarks and future directions.

## **2. Production UNICORE**

UNICORE (UNiform Interface to COmputing RESources) is a vertically integrated Grid system that combines resources of High Performance Computing (HPC) centres and makes them available through the Internet [25]. The UNICORE Grid system emphasizes strong security relying on X.509 certificates, SSL, and the UNICORE User DataBase (UUDB). The security is enforced in a consistent and transparent manner, hiding the differences between platforms from the user thus creating a seamless HPC portal for accessing supercomputers. UNICORE is easily to deploy, and it does not make any assumptions on the administrative and technical environment of a supercomputer centre. Each deploying site decides which resources they will provide. The UNICORE client is a sophisticated and fully-featured Java application. In addition it offers an extension mechanism which facilitates the development of application-specific plugins that further simplify job creation for the user. The UNICORE components are available as Open Source under BSD license on SourceForge [15]. The Research Centre Juelich supports academic users of UNICORE and coordinates its future development. UNICORE is used in many HPC centers in Europe, US and Asia-Pacific, it is the Grid middleware of the EU funded infrastructure project DEISA [6], and the German Grid initiative, named D-Grid [5].

The UNICORE GRID system consists of three distinct software tiers. Firstly, the UNICORE client

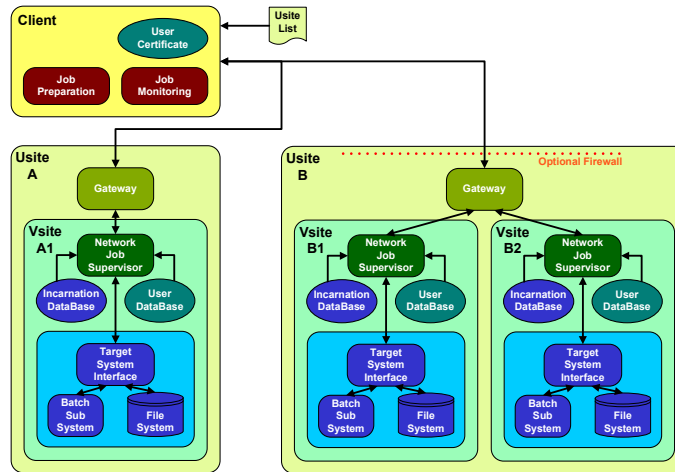


Figure 1. Production UNICORE Grid system architecture

interacts with Grid resources through the sending and receiving of 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.

The second tier are the UNICORE servers that are divided into gateways acting as single points-of-entry into the protected domains of the HPC centres (referred to as USites, see Fig. 1), and Network Job Supervisors (NJSs) that adapt the abstract UNICORE job for the specific HPC system.

The third tier is the target system tier that interfaces with the local resource manager represented for example a batch system like LoadLeveler. The UNICORE Target System Interface (TSI) is the only component that needs to run on the Grid resource - the supercomputer or cluster. Java and Perl implementations of the TSI are available.

The collection of sites which the user configures in the UNICORE client makes up what might be referred to as a personal Virtual Organisation. The UNICORE gateway authenticates the user, before contacting the UNICORE NJS that controls the target system on which the job will be run. The NJS manages the submitted jobs and converts the abstract jobs into batch jobs using the Incarnation Database (IDB), and runs them on the native batch subsystem. Status information and job output are retained by the NJS until the client requests the transfer and deletes the job. The NJS contains a platform-specific incarnation database that adapts to the multitude of current HPC systems. In UNICORE V4.6 the security model is extended to allow for Explicit Trust Delegation (ETD). The user and the site administrators have to explicitly trust a third party to act on behalf of the user [24]. This paper introduces just the basic components of production UNICORE, for an extensive insight and understanding of the whole architecture please refer to [35].

### 3. Early Web services-based prototypes and technologies

In simple cases Web service invocations are stateless; the invocation of the service does not depend on a history of previous interactions. In early Grid prototypes and adapters, the Grid communities re-

alized that the concept to access and manage state across invocations, e.g. Service Level Agreements (SLAs) between Grid resource providers, also plays an important role in Grid computing. First ideas led to stateful entities, physically represented by Grid resources (e.g. Supercomputers or massive storage devices) that manage the state, and allow easier, but not less secure, remote access through defined Grid services that interact with these resources. While Web service implementations were more focussed on business proprietary models that access and manage state related to purchase orders of customer sessions, Grid implementations focussed more on the vision of creating a world-wide interoperable computational Grid. This led to the creation of the Open Grid Services Architecture (OGSA) concepts that are fairly high-level and can not be implemented without a precise specification. Later, both the Web services and Grid community, realized that it is desirable to define Web service conventions for such stateful interactions. These interoperable conventions and resulting emerging standards stem from the discovery of, and introspection on, to the interaction with stateful resources of any type. The work of the Open Grid Services Infrastructure (OGSI) working group at the GGF began to address these issues and to define such interoperable conventions. OGSI defines a set of conventions and extensions for the use of the Web Services Definition Language (WSDL) [22] and XML Schema [16] to enable standardized access and management of stateful Web services. Furthermore, it defines mechanisms of managing the life-cycle of resources and its state that can be accessed via defined operations by Grid services, also named as stateful Web services nowadays.

During the development of the OGSI specification that was finally released in July 2003, many early Grid prototypes were implemented and had contributed to this first process of providing a real specification as a foundation for the higher level OGSA design concepts. These include first versions of the Globus Toolkit that led to the Web service-based reference implementation of OGSI in version 3. In parallel, early Grid service demonstrators were built around the production UNICORE Grid system. These demonstrators and OGSA adaption concepts for UNICORE were mainly developed during the Grid Interoperability Project (GRIP) [2]. Hence, work on UNICORE towards OGSA concepts was motivated by the demands of standards-compliant services that provide access and interoperability to other Grids that deploy their own services, in particular those using the Globus Toolkit. Therefore, developers of UNICORE presented an evolution of UNICORE towards a service-oriented Grid system as a roadmap to achieve a consistent development towards an OGSA implementation in [32]. The concepts of this roadmap included the provision of OGSI-compliant portTypes [22] as well as the design of XML-based protocols, parallelling the UPL/AJO based implementation. The portTypes were modelled following the structure of the UNICORE components, leading to UPL, IDB, UADB, NJS-TSI and Broker portTypes hosted within environments such as webMethods GLUE [9]. These developed portTypes provide a Web service interface for the underlying production UNICORE components and thus decomposing their functionality in essential parts leading to a looser coupling of the components. These concepts were realized by a prototype UNICORE demonstrator that supported a generalized, distributed, application-steering Grid service. More details on this early OGSI-based demonstrator by Fujitsu and its detailed comparison to OGSA concepts can be found in [34]. While the early prototypes and the Globus Toolkit 3 evolved, many Grid applications were starting to be created world-wide along with specifications on top of OGSI within GGF.

While OGSI became the “de-facto” standard for Grid service programming, the Web services architecture evolved, for example by the release of the WS-Addressing [21] specification. This lead to considerations of how the functional capabilities of OGSI can exploit functionality provided by other WS-\* specifications, in particular WS-Addressing. Furthermore, OGSI combined into one specification functionality that is independently useful such as the notification concept. Therefore recently, a refactoring of OGSI has produced five specifications that are named collectively the Web Services

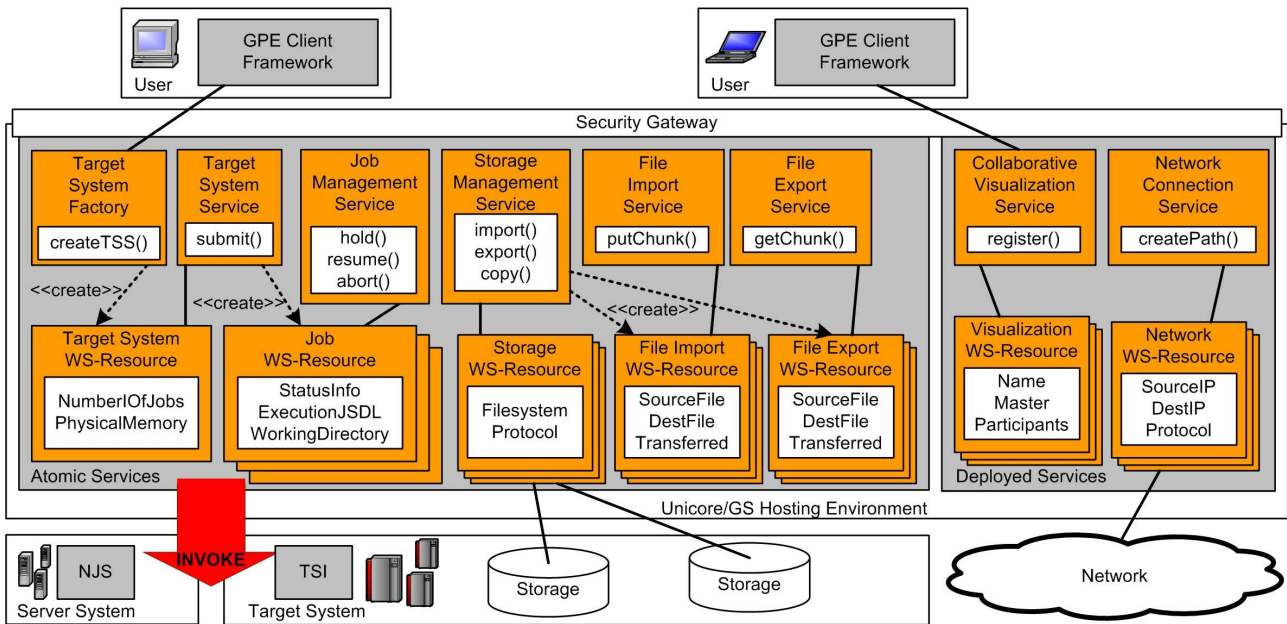
Resource Framework (WS-RF) [23]. In addition, further refactoring led to the definition of three specifications targeting the notification concept of OGSI, named Web Services Notification (WS-N) [30] family. Hence, WS-RF provides comparable functionality to OGSI, but the partitions allows flexible composition capability, and greater reliance on broadly accepted Web service concepts provide a simpler, and incremental path for Web services and Grid developers [28]. In the meantime, GGF specifications that were built on the OGSI specification were refactored towards WS-RF compatibility too, for instance the Web Services Agreement [19] specification.

## **4. Unicore Grid Services Architecture**

Web services technology has proven to be a important foundation for building modern Grid middleware, as demonstrated by new Web services based Grid technologies such as UNICORE/GS and the Globus Toolkit 4 (GT4). While production UNICORE is used in daily production in many HPC centres, the European project UniGrids has driven the further evolution of UNICORE that is compliant with emerging Grid and Web service standards in the context of the OGSA. At the time of printing, OGSA architectures are typically realized by using OASIS WS-RF and W3C WS-Addressing, therefore this next generation of production UNICORE, named Unicore/GS, is also built on these standards. Unicore/GS is designed to be ready-to-use end-to-end solution and thus provides a different approach compared to the toolkit character of Globus, where additional software components are usually needed to integrate Grid functionality into solutions or applications. Unicore/GS is open source and available at SourceForge [15].

### **4.1. Emerging standards compliant Grid services**

OGSA concepts are realised by using the emerging standards WS-RF and several WS-\* technologies as the base infrastructure. The goals of the integration of these standards are two-fold. Firstly new architectures provide a standardized access to Grid resources. We highlight the the UniGrids Atomic Services (shown in Figure 2) that are strongly contributing to standardisation process as a powerful addition to the results of the OGSA - Basic Execution Services (BES) GGF working group. The atomic services define mandatory functionality for system, file and job management within Grids and potentially in other infrastructures such as pervasive computing environments. Secondly, these atomic services facilitate interoperability with other WS-RF compliant middleware. For demonstration purposes, the interoperability between Unicore/GS and GT4 was succesfully shown by the provision of an atomic services implementation for GT4.



**Figure 2. Unicore/GS architecture that facilitates standardized access to Grid resources via the atomic services.**

As illustrated in Figure 2, the atomic services consist of several stateful Web services that allow for the submission and management of computational jobs, easy access to storage resources, and file transfers between systems. Unicore/GS consists not only of a WS-RF implementation, and UNICORE server components as execution backends, but also of a general hosting environment that is capable of hosting other Web services services, including stateful WS-Resources [14]. The implementation of the services and the hosting environment uses the AXIS [31] engine, enhanced to support WS-Addressing, WS-RF and WS-N. Figure 2 also illustrates that the execution backend on the target system is currently realized by an NJS and TSI of the production UNICORE Grid system. Hence, existing well-tested technology is interfaced with WS-\* technologies to provide standardized access. Note that the Japanese National Grid Research Initiative (NAREGI) [12] uses also production UNICORE as a basis for further implementations, because of its well-tested and mature components.

Within Unicore/GS, the UNICORE Gateway component provides the secure transmission of SOAP between a service requestor and endpoint service. In addition, the Gateway is extensible to support other Protocols, e.g. for dedicated streaming services or parallel HTTP that is currently being developed by the European NEXTGrid project [3]. In particular, the default channel for the new Gateway dispatches incoming SOAP messages that understand WS-Addressing headers in the SOAP headers. Also, the new Gateway can use the request URL information from the HTTP transport layer to dispatch SOAP messages which do not include WS-Addressing headers. For large data transfers, streamed data or for interactive applications requiring fast response times and low latencies, the characteristics of SOAP over HTTPS are usually unsatisfactory compared to application-specific proprietary protocols. Therefore, the new Gateway also provides extensions for streaming data without using SOAP and HTTP. The security within Unicore/GS uses X.509 certificates, a standard by IETF. More precisely, every request to a service hosted within Unicore/GS can be realized using transport-level security with X.509 credentials. UNICORE/GS is also considering supporting message-level security using WS-Security [13] although there are some concerns regarding performance implications.

The UUDB provides authorization by mapping X.509 certificates to explicit xlogins and user ac-

counts on the physical resources (e.g. supercomputer). Because of the integration of the ETD [24] mechanisms into the UNICORE server components, also Unicore/GS provides a secure delegation of tasks on the behalf of the user to different Grid sites. Furthermore, it is planned that a UADB service could be possibly shared within a virtual organization [27] and thus allows easier management of authentication and authorization issues in large virtual organizations. This means one centrally managed UADB could exist that stores the needed certificates and CA information for all users of Grid middleware systems within a virtual organization.

The submission and management of computational jobs is handled by the Target System Service (TSS) and the Job Management Service (JMS), which are defined as part of the Unigrids Atomic Services. More precisely, the Target System Factory (TSF) can be used to create a Target System WS-Resource and thus is an implementation of the WS-RF factory pattern, which is defined as any service that is capable of bringing a WS-Resource into existence [14]. This Target System WS-Resource models a physical Grid resource such as a supercomputer or cluster and exposes its state via WS-Resource Properties (WS-RP) [14]. This ranges from the current CPU load, and available physical memory, to the total number of running jobs, processor information, and provided application-specific software. Hence, the Target System WS-Resource can be easily used for monitoring the current status of a system or to provide up-to-date information for remote brokering services. In addition, the TSS provides access to the Target System WS-Resource and defines a *submit()* operation. To submit computational jobs to Unicore/GS, this operation can be used by adding as a parameter a job description using the emerging GGF standard Job Submission Description Language (JSDL) [20]. Internally, the JSDL job descriptions will be mapped to AJOs in Unicore/GS and the real execution will be delegated to the UNICORE server components, in particular the NJS and the underlying TSI. Note that JSDL is a small subset of AJO, but studies into more powerful orchestration and workflow languages such as Business Process and Execution Language (BPEL) [18] is progressing. However, the submitted JSDL will be stored as part of the state of a Job WS-Resource that is finally created by the *submit()* operation and represents a model of the executed job of the target system. The Job WS-Resource also exposes other properties such as the status of the job (e.g. running, queued, or executing), and the working directory of the job. Hence, this information can be used when using Secure Shell (SSH) connections to examine the job output in the working directory during execution, or the start of application-specific software that analyses the execution of the job, for instance debuggers or performance testing tools. The Job WS-Resources can be accessed via the JMS which enables job control operations upon submitted jobs, for instance *hold()*, *resume()*, or *abort()*. Also, the published state of the Job WS-Resources can be read by monitoring services in a standard compliant way and thus allow innovations of easier developments of monitoring services.

So far, we have presented capabilities of Unicore/GS related to job submission and execution. In addition to these capabilities, Unicore/GS provides also several services that are related to storage and file management. In particular, a Storage Management Service (SMS) can be used to access Storage WS-Resources that are automatically created by the TSF on startup of each Unicore/GS service. These Storage WS-Resources expose properties such the underlying filesystem or a list of supported protocols, which indicate the protocols that can be used for file exports/imports (for instance GridFTP, Secure Copy (SCP) or UPL). However, the *import()*, *copy()* and *export()* operations of the SMS can be used to create file import/export WS-Resources and other data management operations on the storage devices. Desirable features of huge file transferal include, in addition to security, monitoring capabilities of the amount of transferred bytes. This is realized by exposing this information within a Import/Export WS-Resource, along with further information such as the source or destination of files. Furthermore, the operations *putChunk()* and *getChunk()* of the File Import/Export Service (FIS/FES)

allow for a transport of files in smaller chunks, operating on the storage devices represented by the correspondent Import/Export WS-Resources.

## **4.2. Client Framework for standardized Grid middleware**

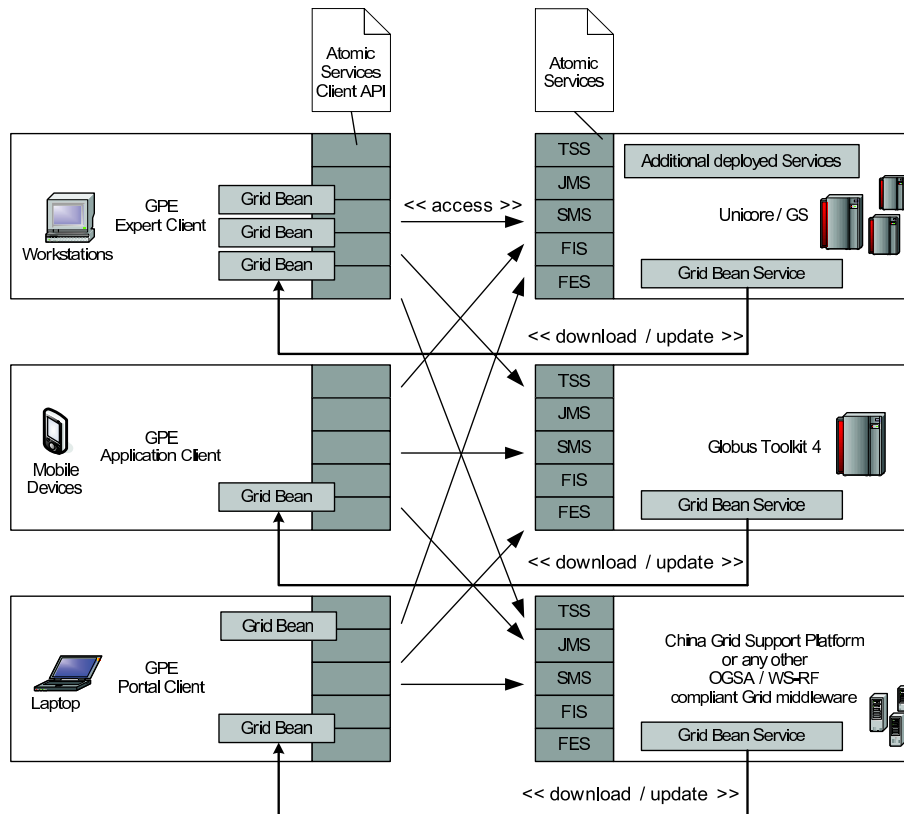
Using a common standardized interface to Grid resources such as the atomic services allows new innovations in creating complex and highly interoperable client frameworks. To provide an example, the job submission or storage management actions, initiated by client users can be requested of a atomic services. Essentially, these Grid resources must provide access to their functionality using the atomic services interfaces as shown in Figure 2.

The Grid Programming Environment (GPE) framework facilitates the usage of the atomic services and is created within the UniGrids project as the client framework for Unicore/GS. In addition, Intel develops additional tools and services that provide a programming toolkit for Grids [10]. GPE offers three different clients for different users. Firstly, an Expert Client that is realized as a Java application running on the users workstation. This client offers all the functionality known from the powerful production UNICORE client, including a workflow-editor, user-identities, and the support for multiple applications. Thus, this client is created to provide full access to Unicore/GS Grids for expert users and administrators. Secondly, an Application Client, a lightweight Java application able to run on mobile devices, offering access to a specific application on the Grid. This client can be very helpful when developing solutions for pervasive computing environments. Finally, GPE provides a Portal client, which is a server application accessible by a web browser. These portals are also developed to be standard compliant with the current Java Specification Request (JSR) 168 that allow for the integration into existing portals like UPortal, Jetspeed or GridSphere.

The GPE also provides a Software Development Kit (SDK) for the development of so-called GridBeans, which are application-specific components similar to the UNICORE client plug-ins in the production UNICORE environment. GridBeans can be run on the different GPE clients without modifications and are thus the interoperable successors of the UNICORE plugins for Unicore/GS environments. The SDK consists of an atomic services client API, libraries and tools that provide the development of portable applications. The implemented GridBeans are deployed via a GridBean service hosted within Unicore/GS. For example, a Grid site offering access to a specific application provides the corresponding GridBean with client functionality for this application. The GridBeans can be easily downloaded and integrated in the GPE clients and allow the access to atomic services implementations on different Grid sites. GPE is also open source [10].

## **4.3. Higher Level Services for Grid Applications**

Figure 2 also depicts two additional deployed services for domain-specific reasons, namely Network Service (NS) and Collaborative Visualization Service (CVS). In modern Grids, such as DEISA or D-GRID, the interconnecting network is a resource of major interest, but often not integrated as services in Grid middleware. Therefore, the NS is used to specify and monitor functional terms (e.g. IP address A, IP address B) and non-functional terms (e.g. required bandwidth, estimated throughput) related to end-to-end connections within Grids. Monitoring of the non-functional terms provides the base for Quality of Service (QoS) checks during the runtime of applications that are using the connection or allows advance reservation of network connections. The negotiation of service characteristics are realized by using the GGF WS-Agreement specification [19]



**Figure 3. Interoperability framework with emerging standard interfaces.**

Another higher level service within Unicore/GS is the CVS, which provides functionality for interactive visualization and computational steering. Furthermore, prototypes are available that allow for the creation of collaborative visualization sessions within Unicore/GS-based Grids. The Visualization WS-Resource provides access to WS-Resource properties such as the name of the visualization and their participants as well as a master role that takes over the management of the visualization session. The data transfer between visualizations and simulations uses SSH and other streaming mechanisms.

Further higher level services will evolve over time, related to science and probably also to business. Note, this is an area which there is lots of new developments.

#### **4.4. Innovations by three-layer interoperability**

An interesting relationship exists between the GPE client framework and the Unicore/GS Grid system. This relationship is represented by an interface that is going to be standardized through the GGF as a powerful addition to the OGSA-BES initiative. The innovative character of this new design of a uniform interface to Grid services, called atomic services, allow completely new forms of interoperability between Grid middleware. For example, the interoperability between production UNICORE and the Globus Toolkit 2 is realized by a so-called GlobusTSI that delegates all commands initiated by the UNICORE user to a Grid Resource Allocation Manager (GRAM) [26] server of Globus. In particular, the GlobusTSI is used instead of the usual TSIs on the server-side and transforms incoming UPL requests into the proprietary Resource Selection Language (RSL) of Globus. The implementation of the production UNICORE GlobusTSI base upon a version of a Commodity Grid (CoG) Kit [4] API that allows only access to Globus Toolkit 2 systems, and not to systems with version

3 or 4 of the toolkit. This also shows the common problems when working with non-standardized adapter approaches and proprietary protocols instead of using one standardized interface, accessible via standardized protocols. One of the major problems of the GlobusTSI is its performance. This means complex workflows that are using different Grid nodes managed by Globus and UNICORE middleware are processed through the whole server hierarchy of UNICORE, including authentication checks at the gateway, authorization checks and job incarnation [35] at the NJS, before one part of the workflow reach the GRAM server. This GRAM server in turn, does the same in the Globus style again (authentication and authorization via Gridmap file [26]), leading to a job response time that can be significantly improved. The upcoming Web and Grid standards will allow new approaches to interoperability. These improvements are realized through the use of the innovative atomic services interface with Unicore/GS and an atomic services implementation for the Globus Toolkit 4. The implementation of the atomic services is not only possible for the Unicore/GS and Globus systems but also for other Grid middleware such as gLite or the China Grid Support Package (CGSP). In fact, the CGSP already support an implementation of the atomic services. Furthermore, complex workflows and GPE client users benefit from the opportunity to access several Grid resources with their favorite client without knowing if a system is managed by developed using Unicore/GS or Globus. This is accomplished by the integration of the atomic services in Unicore/GS or in Globus solutions. The change of the Grid system that provides access to a Grid resource is completely transparent to the GPE users, as long as the Grid systems implement the atomic services.

In addition, the standardized access to properties of Target System WS-Resources and Job WS-Resources (e.g. number of jobs, or job status information) allow new developments for Grid broker, which until now they only provide adapters to the correspondend protocols of the underlying Grid middleware (e.g. RSL, UPL) and are thus not scalable and difficult to keep up-to-date with the latest versions of the protocols.

Furthermore, the definition of such atomic services and their standardization efforts allow new possibilities within pervasive computing, realizing the computing everywhere at anytime with a specific interface to perform huge computational tasks within Grids that modern mobile devices (e.g. PDAs, or smart phones) can not achieve. This becomes particularly useful when anticipating user actions in pervasive computing environments, to provide up-to-date information in real-time to the user of the correspondend pervasive computing application. Hence, by the use of the atomic services, the development of applications for pervasive computing can be made easier and more alligned to the Grid computing environments. Additionally, self-management capabilities as stated in [33] can be realized when committing to an emerging negotiation standard such as WS-Agreement [21].

To sum up, the atomic services interface in combination with the GPE client framework provide three levels of interoperability. Firstly, it provides pure WS-RF interoperability which was successfully tested with Unicore/GS at the WS-RF interop fest, which is held regularly to check standard compliance of systems. Secondly, the interoperability between the different atomic services implementations and their GPE client API that hides the details about implementation issues of the correspondent Grid middleware. The third level of interoperability is achieved by the GridBeans that are working on top of the three different client implementations and can access Unicore/GS or Globus Toolkit based systems without any modifications. Hence, this three layers of interoperability allows for the development of GridBeans with application-specific code that survive version changes in the underlying layers and are thus very easy to maintain. In particular, the application-specific code inside the Grid-Bean is independent of the atomic services specification, and independent of WS-RF specifications and is therefore considered to be a long living solution for Grid applications and usage scenarios. In

conclusion, the Unicore/GS and its GPE client architecture can be seen as an enormous innovation in Grids today, mainly driven by its commitment to emerging standards.

## 5. Conclusions

We reported here on the progress of the atomic services interfaces and their Web service-based reference implementation Unicore/GS that can be seen as an innovation towards effective interoperability within Grids. This innovative interoperability allows for new broker, monitoring or workflow services that will evolve over time. The upcoming standardization efforts of the atomic services as additions to the already initiated OGSA-BES GGF working group plays a key role in modern Grid computing. Unicore/GS massively commits to emerging standards like W3C WS-Addressing, IETF X.509 certificates, OASIS WS-RF and WS-N, and several GGF specifications, e.g. JSDL, OGSA-BES, and WS-Agreement. This commitment plays a major role in Grid computing and allows for the use of Grids by new applications areas such as pervasive computing, or the creation of new massive complex workflow tasks that need a wide range of interoperability in Grids without loosing performance. The progress on standardization activities is more and more evolving and the standardization processes of modern Grid systems like Unicore/GS are contributing to the vision of Grid Computing: a world-wide Grid that is easy accessible and a secure interoperable IT infrastructure. Standardization brings this vision closer to reality and production environments. We presented early OGSi-based prototypes and their evolution towards Unicore/GS by using WS-RF. Unicore/GS is not a prototype anymore and can be soon used in production. DEISA and D-Grid communities have taken keen interest in Unicore/GS. The german company T-Systems Sfr provides professional support for Unicore/GS and integrated it in their testing and release-management systems. *While production UNICORE is succesfully used in daily production at many HPC centers world-wide today, we presented here that Unicore/GS has the potential to improve it in terms of interoperability, usability, standards compliance, and functionality.*

## References

- [1] European UniGrids Project, <http://www.unigrids.org>.
- [2] European GRid Interoperability Project, GRIP, <http://www.grid-interoperability.org>.
- [3] European NEXTGrid Project: Architecture for Next Generation Grids, <http://www.nextgrid.org/>.
- [4] Commodity grid (cog) kits. <http://www.cogkit.org/user>.
- [5] D-Grid Initiative - German Grid Initiative. <http://www.d-grid.de>.
- [6] DEISA - Distributed European Infrastructure for Supercomputing Applications. <http://www.deisa.org>.
- [7] gLite middleware of the EGEE project. <http://public.eu-egee.org/>.
- [8] Global Grid Forum. <http://www.gridforum.org/>.
- [9] GLUE. <http://www.webmethods.com/meta/default/folder/0000008629>.
- [10] Grid Programming Environment. <http://gpe4gtk.sourceforge.net/>.
- [11] Internet Engineering TaskForce. <http://www.ietf.org/>.

- [12] NAREGI - Project. <http://www.naregi.org>.
- [13] OASIS - WS-Security TC. [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss).
- [14] OASIS - WSRF TC. [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrf](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf).
- [15] UNICORE at SourceForge. <http://unicore.sourceforge.net>.
- [16] XML Schema Definition Language, W3C. <http://www.w3.org/XML/Schema>.
- [17] W. Allcock. GridFTP: Protocol Extensions to FTP for the Grid. <http://www.ggf.org/documents/GFD.20.pdf>.
- [18] T. Andrews et al. Business Process Execution Language for Web Services, 2003. Version 1.1.
- [19] A. Andrieux et al. Web Services Agreement Specification, 2004. Version 1.1, draft 20.
- [20] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language Specification. <http://www.ggf.org/documents/GFD.56.pdf>.
- [21] D. Box et al. Web Services Addressing (WS-Addressing), 2004. <http://www.w3.org/Submission/ws-addressing/>.
- [22] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL), 2001. [www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl).
- [23] K. Czajkowski et al. The Web Services Resource Framework. <http://www.oasis-open.org/committees/download.php/6796/ws-wsrf.pdf>.
- [24] V. Li D. Snelling, S. van den Berghe. Explicit trust delegation: Security for dynamic grids. *FUJITSU Scientific and Technical Journal*, 40(2):282–294, December 2004. <http://www.fujitsu.com/global/news/publications/periodicals/fstj/archives/vol40-2.html>.
- [25] D. Erwin, editor. *UNICORE Plus Final Report*. UNICORE Forum e.V., 2003.
- [26] I. Foster. Globus Toolkit 4: Software for Service-Oriented Systems. In *IFIP International Federation for Information Processing, LNCS 3779*, 2005.
- [27] I. Foster, C. Kesselmann, J. M. Nick, and S. Tuecke. *The Physiology of the Grid*, pages 217–249. John Wiley & Sons Ltd, 2003.
- [28] I. Foster, D. Snelling, et al. Modeling and Managing State in Distributed Systems: The Role of OGSF and WSRF. In *Proceedings of the IEEE, Vol.93, No.3*, 2005.
- [29] Ian Foster et al. Open Grid Services Architecture. <http://www.ggf.org/documents/GFD.30.pdf>.
- [30] S. Graham, J. Treadwell, et al. Web Services Notification, 2004. <http://docs.oasis-open.org/wsrf/2004/11/wsrf-WS-Notification-1.1-draft-01.pdf>.
- [31] Apache Group. Apache eXtensible Interaction System. <http://ws.apache.org/axis/>.
- [32] 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)*, pages 142–150, 2003.

- [33] M.Riedel, V.Sander, and M.Fidler. Self-Management Functions in WS-Agreement-based Grids. In *Proceedings of the Autonomic Grid and Networking Workshop, MANWEEK05, Barcelona*.
- [34] 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.
- [35] 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, 2005. Elsevier, L. Grandinetti (Edt.), *Grid Comp. and New Frontiers of High Performance Proc.*
- [36] S. Tuecke, K. Czajkowski, I. Foster, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open Grid Services Infrastructure. <http://www.ggf.org/documents/GFD.15.pdf>.