Concepts and Design of an Interoperability Reference Model for Scientific- and Grid Computing Infrastructures

Morris Riedel, Achim Streit, Thomas Lippert, Felix Wolf
Juelich Supercomputing Centre
Forschungszentrum Juelich
Wilhelm-Johnen-Str. 1, D-52425 Juelich
Germany
m.riedel@fz-juelich.de

Dieter Kranzlmueller
Department for Informatics
Ludwig Maximillians University Munich
Oettingenstr. 67, D-80538 München
Germany

Abstract: - Many production Grid and e-science infrastructures offer their broad range of resources via services to end-users during the past several years with an increasing number of scientific applications that require access to a wide variety of resources and services in multiple Grids. But the vision of world-wide federated Grid infrastructures in analogy to the electrical power Grid is still not seamlessly provided today. This is partly due to the fact, that Grids provide a much more variety of services (job management, data management, data transfer, etc.) in comparison with the electrical power Grid, but also the emerging open standards are still partly to be improved in terms of production usage. This paper points exactly to these improvements with a well-defined design of an infrastructure interoperability reference model that is based on open standards that are refined with experience gained by production Grid interoperability use cases. This contribution gives insights into the core building blocks in general, but focuses significantly on the computing building blocks of the reference model in particular.

Key-Words: Scientific Computing, Grid Computing, HPC, HTC, Interoperability

1 Introduction

Computational simulations and thus scientific computing is the third pillar alongside theory and experiment in science and engineering today. The term e-science evolved as a new research field that focus on collaboration in key areas of science using next generation computing infrastructures such as Grids to extend the potential of scientific computing.

More recently, increasing complexity of e-science applications that embrace multiple physical models (i.e. multi-physics) and consider a larger range of scales (i.e. multi-scale) is creating a steadily growing demand for world-wide interoperable Grid infrastructures that allow for new innovative types of e-science by jointly using a broader variety of computational resources. Since such interoperable Grid infrastructures are still not seamlessly provided today, the topic 'Grid interoperability' emerged as a broader research field in the last couple of years.

The lack of Grid interoperability is a hindrance since we observe a growing interest in the coordinated use of more than one Grid with a single client that controls interoperable components deployed in different Grid infrastructures. In fact, we have shown in a recent classification [10] that among simple scripts with limited control functionality (i.e. loops), scientific application client plug-ins, complex workflows, and interactive access, there is also Grid interoperability mentioned as one approach to perform e-science today.

Such interoperable federated Grids have the potential to facilitate e-research and thus scientific advances, which would not be possible using only a single Grid infrastructure. These advances arise from the advantages that federated Grid resources provide, such as access to a wide variety of heterogeneous resources, aggregated higher throughput, and lower time-to-solution.

In more detail, we observe that more and more Grid end-users raise the demand to access both High Throughput Computing (HTC)-driven Grids (EGEE, OSG, etc.) and High Performance Computing (HPC)-driven infrastructures (DEISA, TeraGrid, etc.) from a single client or science portal. In this context, the fundamental difference between HPC and HTC is that HPC resources (i.e. supercomputers) provide a good interconnection of cpus/cores while HTC resources (i.e. pc pools) do not.

This joint use is typically motivated by the theory and concept that tackle the scientific problem and is modeled within the corresponding codes that in turn lead to some codes that are 'nicely parallel' (i.e. HTC) and others that are better suited to be computed as 'massively parallel' (i.e. HPC) simulations. In addition, the joint use of HTC- and HPC-Grids is often motivated by the fact that often end-users perform smaller evaluation runs with their codes on HTC resources before performing full-blown production runs on large-scale HPC resources. This saves rare computational time on costly HPC resources within HPC-driven Grids.

This contribution highlights recent achievements in defining the core building blocks of a Grid infrastructure interoperability reference model (IIRM) that is based on emerging open standards and there improvements based on lessons learned from real interoperability use cases. Although many aspects of the reference model have been described by Riedel et al in [3], this paper provides much more detail in the particular context of the core building blocks in terms of computation. In this sense it highlights the improvements of the emerging open standards that have proven to be useful in production Grids. In fact, more recently, the most elements of the work provided within this contribution have been given as an input into the OGF Production Grid Infrastructure (PGI) working group in order to feed back our lessons learned and interoperability application experience with open standards into the standardization process.

This paper is structured as follows. Following the introduction, Section 2 sets the scene by discussing the common challenges and benefits of Grid interoperability. Section 3 describes the general design of the interoperability reference model, while Section 4 provides many details to the improved concepts of open standards. Finally, after surveying related work in Section 5, we present our conclusions in Section 6.

2 World-wide Grid Islands

At the time of writing, it is an interesting time period for European Grids in the sense of the upcoming transition process from the project-based EGEE project to a longer sustainable European Grid Initiative (EGI) while DEISA and the Partnership for Advanced Computing in Europe (PRACE) are jointly creating an HPC infrastructure for emerging peta-scale applications. In the US, we see an upcoming third phase of the TeraGrid in the context of the extreme digital (XD) resources for science and engineering transition.

Nevertheless, what we learned from the past and what we can expect from the future is that the underlying computing paradigms will remain in the sense that requirements for HTC and HPC will be still present. That's still valid even in times where, in principle, HTC and HPC codes could be executed on one large-scale cluster such as the IBM BlueGene/P while having thus much more focus on the computed data itself instead on the computational paradigms that are being used. More recently this approach have been coined as many-task computing that is rather close to the approach of Grid interoperability in the sense of using HTC and HPC concepts seamlessly and focusing much more on the data aspect of the scientific applications.

Since the difference between these underlying computational paradigms (i.e. HTC and HPC) will still exist in the future, interoperability between Grid infrastructures that offers seamless access to both types of computational resources will be further needed in future. Since, we observe a rather slow adoption of emerging open standards in deployed Grid middleware systems on these infrastructures in the past; The Grid communities and projects developed many different approaches to Grid interoperability that are classified by Riedel et al. in [10].

What we observe in all of these approaches, that are not only restricted to HTC and HPC infrastructure interoperability, is that emerging open standards have a high potential to achieve a reasonable basic level of interoperability. But in terms of production use cases it turns out that many standards that are adopted rather slowly in production Grids lack some certain smaller concepts. Often, the interoperability is achieved by doing small workarounds; apply small hacks or changes to the emerging standards to get not-fully supported concepts working on a pair-to-pair basis between usually two production Grid infrastructures.

As a summary, common open standards are the one and only way to enable a long-term seamless cross-Grid access that goes beyond a pair-to-pair basis connecting some of the so-called non-interoperable 'Grid islands'. This, however, implies that experience gained in production Grid interoperability must be fed back to the standardization process. Thus we have worked on the understanding of how such standards can be further improved to increase their adoption in production Grid middleware.

In fact, we have performed many interoperability tests and worked with a wide variety of interoperability use cases [2] in the context of the Grid Interoperation Now (GIN) community group of OGF. The lessons learned from all these activities have been given as an input to the OGF Production Grid Infrastructure (PGI) working group in order to improve existing emerging open standards towards an improved production usage following the well defined infrastructure interoperability reference model [3]. In this working group, members of UNICORE (deployed on DEISA), ARC (deployed on NorduGrid), and gLite (deployed on EGEE) work closely together with members of the US (e.g. GENESIS-II) in order to define how these improved standards can be seamlessly integrated into middleware.

3 Reference Model Design

The lack of Grid interoperability is a hindrance since we observe a growing interest in the coordinated use of more than one Grid infrastructure from a single Grid client, which is able to seamlessly use a variety of computational resources in different Grid infrastructures.

The fundamental idea of the Grid infrastructure interoperability reference model (IIRM) is to formulate a well-defined set of emerging open standards and refinements of them in order to address the interoperability needs of state-of-the-art production Grid and e-science infrastructures. In order to identify this well-defined set of standards, we worked with many scientific interoperability use cases [1, 2]. Based on these efforts, the lessons learned about the most crucial functionality led to the core building blocks of the reference model design as shown in Figure 1.

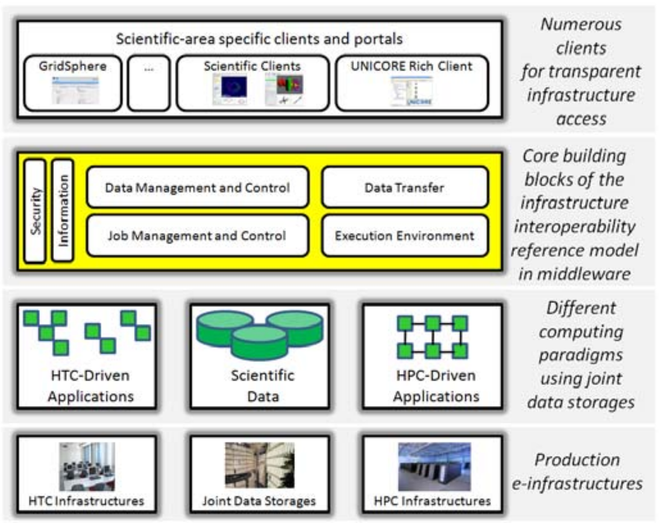
Scientific-area specific clients and portals
GridSphere
...
Scientific Clients
UNICORE Rich Client
Numerous clients for transparent infrastructure access
Security
Information
Data Management and Control
Data Transfer
Job Management and Control
Execution Environment
Core building blocks of the infrastructure interoperability reference model in middleware
HTC-Driven Applications
Scientific Data
HPC-Driven Applications
Different computing paradigms using joint data storages
HTC Infrastructures
Joint Data Storages
HPC Infrastructures
Production e-infrastructures

Fig.1 – The core building blocks of the infrastructure interoperability reference model in the context of state-of-the art scientific and Grid computing infrastructures.

As shown in Fig. 1, the core building blocks are well embedded in the typical environments of Grid infrastructures with numerous types of clients accessing them in order to execute different types of applications that are typically based on different computing paradigms (i.e. HTC, HPC) using some kind of shared scientific data. In the most cases, different production Grid infrastructures exist to satisfy these demands that are HPC-driven Grid infrastructures (i.e. TeraGrid, DEISA) with large-scale HPC resources and HTC-driven Grids (i.e. OSG, EGEE) with a high amount of smaller clusters or PC pools.

In addition, in many applications use cases we encountered the demand for joint data storages since data is fundamentally different from computation in the sense that data once stored in one technology must be migrated in a time-consuming effort. In computing, on the other hand, the submission and management of the computational jobs itself can be easier changed to another infrastructure although also this implies certain problems (e.g. differences in job description).

Hence, the identified most crucial functionality to actually enable interoperability between production Grid infrastructures is data management and control, including the data transfer, as well as job management and control. In context of the latter, there is also the execution environment important, for instance to have common environment variables that describe different boundary conditions (e.g. available memory, CPUs, etc.) for the application execution during run-time.

Apart from this functionality, there are two special kinds of elements of the design that are security and information. A common security setup is in the most cases the major showstopper to enable interoperability between Grid infrastructures and as it affects basically every layer in can be considered as a rather vertical building lock (cp. Fig. 1). The same is valid for information that refers to the up-to-date information about the computational Grid in general and each of its computational resources in particular. These pieces of information are reaching from the amount of available CPUs to a list of supported applications and services.

Less used in our interoperability use cases have been self-management functionality or service level agreements that are often used in other use cases together with a meta-scheduling framework. Investigations in production Grid infrastructures reveal that these frameworks as well as self-management functionality is rather experimental and not used in production Grids on a daily basis.

Further investigations [2] in production Grids reveal that the most interoperability use cases are often based on emerging open standards including some modifications and thus small refinements of them. By taking our various lessons learned into account, these open standards (including refinements) can be in turn easily mapped to the core building blocks of the reference model design as shown in Fig. 2. In this context, we refer to profiling in the sense of defining small refinements of the used open standards that have been proved useful in numerous interoperability use cases and thus majorly improve the effectiveness of the emerging open standard.

As shown in Fig. 2, the significant improved standards are the Storage Resource Manager (SRM) [4] data control interface as well as the WS-Database Access and Integration Service (WS-DAIS) [5] data management interface. In terms of wide-area data transfer, mostly GridFTP or HTTP(S) is used in the scientific use cases. The core building block of job management and control is represented by improved versions of the OGSA-Basic Execution Service (BES) [6] and the Job Submission and Description Language (JSDL) [7]. The environment profile is somehow different since its standardization was started within the GIN community group, but not yet finalized as proposed standardization document. The information standard that plays a very important role in the design model is the GLUE2 [8] standard including some small refinements. Finally, the security profile refers to a broader range of authentication and attribute-based authorization standards basically based on X.509 and the Security Assertion Markup Language (SAML) [9].

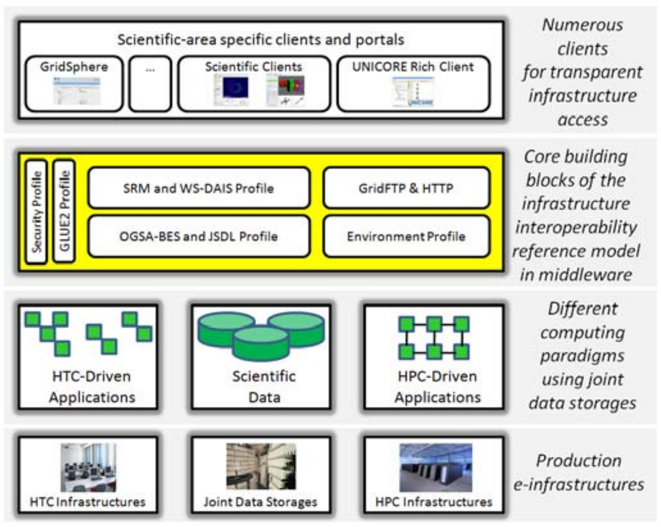
Scientific-area specific clients and portals
GridSphere
...
Scientific Clients
UNICORE Rich Client
Numerous clients for transparent infrastructure access
Security Profile
GLUE2 Profile
SRM and WS-DAIS Profile
GridFTP & HTTP
OGSA-BES and JSDL Profile
Environment Profile
Core building blocks of the infrastructure interoperability reference model in middleware
HTC-Driven Applications
Scientific Data
HPC-Driven Applications
Different computing paradigms using joint data storages
HTC Infrastructures
Joint Data Storages
HPC Infrastructures
Production e-infrastructures

Fig. 2 – The reference model core building blocks are based on refined open standard profiles.

Because of the page limitation, we are not able to cover all the details of all the core building blocks. Therefore, this contribution focuses on the OGSA-BES and JSDL profile building block including some aspects of the environment profile as well. For more overall pieces of information about the other building blocks of the reference model please refer to Riedel et al. [3].

4 Computational Concepts

The whole IIRM is based on lessons learned gained from using the emerging open standards in real production e-science interoperability applications. In this context, the paper focuses only on the improvements related to the emerging standards OGSA-BES and JSDL while an overall description of the reference model itself can be found in [3].

We gained a lot of experience in the past several years with using the OGSA-BES service implementation not only for pure demonstration purposes, but also for real interoperability use cases that require resources in more than one Grid infrastructure. While working with these different Grid applications, we encountered several limits in using the OGSA-BES specification in several scientifically-driven interoperability use cases among several infrastructures. These limitations have been addressed by many different concepts that lead to the proposed improvements of the emerging open standards that we describe within this section in more detail.

When comparing these improved concepts, we outline that the OGSA-BES specification concepts do not fit the requirements of production e-science infrastructure job management as experienced by use cases such. This is also true when combining it with profiles such as the High Performance Computing (HPC) – Basic Profile (BP) that also not cover all of the following concepts. We thus argue that the scope of the OGSA-BES specification must be extended in the case of scientifically driven Grid infrastructures even if this means a decrease in modular approaches of reusing the OGSA-BES specification in numerous other use cases.

In fact, the broad variety of how the OGSA-BES specification can be used with numerous other security setups and profiles lead in several of our application use cases to a significant decrease in successful interoperability setups. Therefore the goal of the IIRM in general, and the improvements of OGSA-BES in particular is to profile exactly how this improved OGSA-BES can be used in the context of deployed components for information handling, data transfer, and storage management as well as security.

Taking the lessons learned of many problems in interoperability into account, it becomes clear that job, data, and storage management can be seen as one atomic entity as initially proposed by Riedel et al. in [18] described by using the UNICORE Atomic Services (UAS) as reference implementation at that time. The UAS provide Web service interfaces for all these parts, but are rather proprietary compared to the OGSA-BES, SRM, and other specifications. Nevertheless, the approach of specifying all these different areas together is the approach we found majorly important while dealing with multi-infrastructure e-science applications. In other words, if one little element is slightly different specified and implemented (e.g. one JSDL-based extension is supported in e-research tool A, but not in e-research tool B) it has a high potential to break the whole interoperability setup.

Also important is to specify at least one output mechanism that acts as default in the improved OGSA-BES specification to avoid the problem of specifying a raw interface that can be used to submit jobs, but not getting any results. In this context, the output transfer is tightly coupled with storage functionality and thus also part of the ’atomic entity’ mentioned above. As a side remark, the ’atomic’ term is not meant as ’atomic transaction’ but rather indicating that job, data, and storage management must be closely defined together as within the IIRM to enable interoperability usable for production applications. Hence, core job submission, storage, and file transfer functionality must be seen as one atomic entity allowing possible ways of extending (e.g. alternative data-staging technologies/profiles).

4.1 Improving Common Open Standards

This section discusses the identified concepts that directly affect the execution service interface and thus our proposed improvements of the OGSA-BES interface. For each of the concepts we also reveal the context of our work with e-research applications that require resources in more than one e-science infrastructure and in turn substantially contributed to our proposed improvements.

In general, we can state that the OGSA-BES specification is very good first step towards the right direction. This is acknowledged where several required concepts are actually already provided by this specification. To provide an example, in the WISDOM [1] and EUFORIA [19] use case, we actually succeeded in submitting a job from the EGEE infrastructure to the DEISA infrastructure and vice versa using the OGSA-BES interface of gLite and UNICORE. We have been able to submit simple jobs while simple refers to the fact of having one executable without very specific resource requirements (e.g. used network topology on a resource).

Also, the cancellation of submitted jobs and their status retrieval also worked fine across the different infrastructures. An interesting concept provided by the OGSA-BES specification was the remote management features that have been marked as deprecated in the improvements since we learned that administrators in production infrastructures typically would like to retain local resource control and are not in favor of service management operations that can be remotely invoked. But despite several advances in interoperability, we also have to state that there are a lot required concepts that are not in OGSA-BES or JSDL adoptions today.

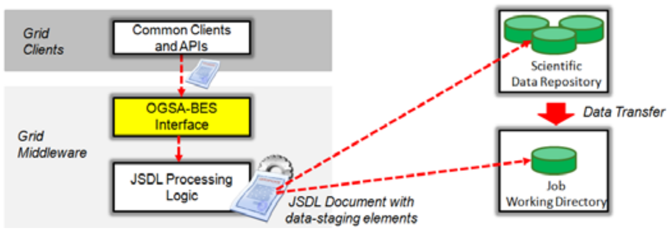

Fig. 3 – OGSA-BES / JSDL defines functionality for staging data automatically performed via the middleware.

One example for the most important missing concept initially originated from the WISDOM use case and we refer to this concept as *client-initiated data-staging*. In the context of this application, it is important to understand that exactly between the workflow step 1 (molecular docking) and step 2 (molecular dynamics, i.e. simulation over time) manual intervention is necessary by the scientists in order to evaluate which docking data actually makes sense to be computed using highly costly supercomputing time. Hence, the scientists manually analyzes the data and afterwards raise the requirement for our an approach that enables ’client initiated data-staging’ to one specific site where a particular job should be executed.

As shown in Fig. 3, as part of JSDL, end-users are able to specify which data should be staged-in by the Grid middleware and we refer to this approach as ’data pull’. Nevertheless, in many of our use cases, there was a specific need by scientists that data has to be staged-in manually to the working space of the corresponding Grid job (i.e. activity). In one particular example, the WISDOM scientists actually would like to submit a job to the UNICORE-BES implementation and before the activity is being started, the scientists use meta-data stored in a WS-DAIS-compliant database and GridFTP to trigger an client-initiated data-staging that only transfers a specific data subset of workflow step 1 (molecular docking) outcomes for computation on the DEISA infrastructure.

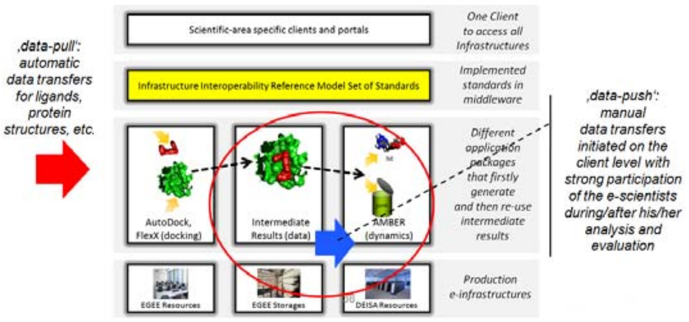
‚data-pull': automatic data transfers for ligands, protein structures, etc.
Scientific-area specific clients and portals
One Client to access all Infrastructures
Infrastructure Interoperability Reference Model Set of Standards
Implemented standards in middleware
AutoDock, FlexX (docking)
Intermediate Results (data)
AMBER (dynamics)
Different application packages that firstly generate and then re-use intermediate results
‚data-push': manual data transfers initiated on the client level with strong participation of the e-scientists during/after his/her analysis and evaluation
EGEE Resources
EGEE Storages
DEISA Resources
Production e-infrastructures

Fig. 4 – Data-push as improvement concept versus already existing data-pull in the context of a real interoperability use case application using the EGEE and DEISA infrastructure.

This particular concept and thus the identified improvement of the standards is shown in Fig. 5 (blue text and lines). More information about this particular e-science application with data-staging using relational databases can be found in Holl et al. [20].

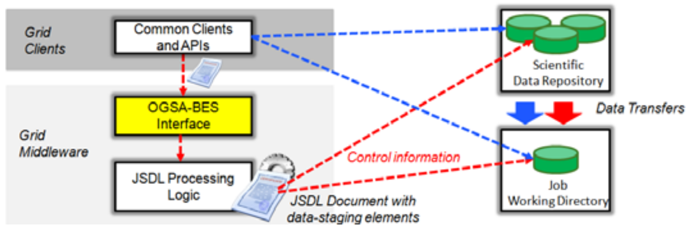

Fig. 5 – Improved OGSA-BES / JSDL defines functionality for staging data manually performed via the client.

A closer investigation of this concept reveals that two more related concepts are actually required to achieve this. First, this concept raises the demand for the concept of *job working directory access*, which refers to the client knowledge of the location of the job sandbox (i.e. directory) where the submitted job will be executed. Second, the submitted job should not start until the client-initiated data-staging has been performed and thus this leads to the concept of *Predefined Hold points*. If this is not provided, the scientists manually transfer sub-elements of the data into the file system GPFS of DEISA and UNICORE-BES in turn would have to stage-in the data for computation into the working directory thus leading to a second not necessary file transfer.

All these three concepts are at the time of writing not supported by the OGSA-BES specification, but can be easily achieved by providing the job location, if possible, as an immediate result of the submit operation (i.e. createActivity) or providing a suitable way of obtaining this location via a query about job information later. This location is important so that end-users (or corresponding client software) knows where data can be staged close to the created Grid job. Of course, this location can be also retrieved with another operation that queries information about one particular Grid job, but we found that this then requires yet another communication exchange and thus was considered to be more efficient when the location information can be encoded in the outcome of a job submission (if possible). Apart from the WISDOM use case, we learned that also other use cases such as EUFORIA, or VPH [3] welcome this concept in order to use this location to perform a direct SSH connection into the working directory. This has been proved to be useful in order to enable higher-level concepts such as computational steering of Grid jobs or checking the data-writing progress of one Grid job while running.

Another aspect to realize the 'client-initiated data-staging'ins the concept of using 'Predefined Hold points' that can be described with 'states'. This means that after a Grid job is submitted and its JSDL has been parsed and the implied data-staging activities have been performed, the job should not directly start in order to wait for any kind of client-initiated data-stagings. In order to communicate this, scientists can specify so-called hold points in the improvements of the JSDL-based job submission. The job description is the right place for specifying this since the position of the hold-points are often related to the nature of the execution itself while a more general and flexible manual suspend functionality is defined in the context of the interface itself.

In the WISDOM example and in the context of workflow step 2 (molecular dynamics), one hold-point is always at the data-staging-in state in order to allow that WISDOM scientists can transfer a suitable subset of data from the molecular dockings before the job is actually starting to run in DEISA. We also learned from the WISDOM use case that from time to time a computational job should be just suspended, which can be also achieved by providing another hold-point during the 'job-execution' state. However, this is not clearly directly related to the job itself and thus this concept of is rather implemented on the execution service interface level. This concept enables, for instance in our WISDOM use case, an evaluation of the already simulated molecular dynamics time frame in order to check if 'invalid movements' might already occur and thus make it unnecessary to continue the simulation. In the most cases this was related to the fact that sometimes computed data can be easily analyzed by the e-scientists that in turn immediately decide whether the data is useful or was just another evaluation run that should be not transferred in data-staging activities to more permanent storages.

The above discussion about 'suspend' leads to a more generic discussion when observing the necessary 'resume' functionality and maybe even other rather manual state changes. In this context, a concept that was typically missing in interoperability use cases was thus the *Manual manipulation of job states* concept, which can be implemented with an operation such as changeActivity() within the execution service interface. But this particular operation is non-trivial since it raises several concerns and was left out initially in the OGSA-BES specification for reasons related to the extensibility of the implied job status model. Hence, additionally supported states might not be known by clients and thus invoking such an operation with an unknown state model might cause serious trouble, for instance the uncertainty in picking the correct state to move an activity to.

We address this issue within the IIRM by explicitly profiling allowed state changes and defining a state model that is sufficient, but still extensible like the one in the OGSA-BES specification. Nevertheless, we encountered several times, not only in the WISDOM, EUFORIA or VPH application that this operation should be provided by an improved OGSA-BES specification. The benefit of this operation is twofold. First, it provides an operation that allows e-scientists to explicit start an activity once the potentially manually performed data-staging step is finished. It thus also plays a crucial role in the 'client-initiated data-staging' concept stated above in order to enable e-scientists with the possibility to finish the data-staging when they want. Second, it provides the functionality to suspend or resume a running Grid job in general, although this raises the requirement that underlying resource management systems support these feature, which was not always the case in our experiences. In this context, the GLUE2-based information service that provides information about any improved OGSA-BES endpoint must expose information whether this feature is supported or not.

This operation demands a well-defined fine-granular state model, which also includes our concept of *Data staging in state model*. In the WISDOM example, in the concept of workflow step 1 (molecular docking), a lot of initial input data (ligands, proteins, etc.) are staged-in before the job actually starts. However, the e-scientists often did not recognize why the job is not running was related to the fact of a long-lasting data-staging-in activity. There is currently no mechanism in the state model in the OGSABES specification that indicates that a data-staging activity is currently performed by the service implementation and thus we propose this as one of the concepts that are required. In a more broader sense, it is important to understand that already small differences even in the state terms (e.g. finished vs. ready) several times broke our interoperability setups or majorly influenced its success. We thus argue that a list of commonly agreed state terms between the different infrastructures and middleware providers is crucial and often its value is underexpected or oversimplified with respect to state extensions that cause serious problems.

Apart from the WISDOM use case we also encountered the need to completely wipe out an activity that include removing all temporary files and other resources allocated to the correspondent Grid job. This concept is not defined in the OGSA-BES specification so far, but required in several interoperability use cases and thus considered in the improved OGSA-BES specification.

One particular use case is to use this concept to clear all storages related to data-staging activities as well as all evidence of the submission. Note the difference compared to the concept of terminating or cancelling Grid jobs where the activity still exists in the services in a suitable terminated or cancelled state. In contrast, the concept of *Wipe-out of submitted jobs* means that the job is not longer available in the Web service container.

5 Related Work

There is a wide variety of approaches in related work in the field of Grid interoperability that we list here. The most of these approaches don't use the approach of improving open standards and as such rather implement 'transformation logic' in one way or the other. This transformation logic is responsible to translate a protocol A into protocol B or a schema A into schema B. This process is typically very time consuming and error-prone.

Furthermore, it is very difficult to maintain since if one element is changed different versions of these transformation logics have to be maintained. In addition, often it implies that a protocol is not fully able to being mapped to another. Hence, the result of transform protocol A into protocol B might actually lead to a protocol B* that is often only a subset of protocol B.

The most famous and thus most common approach is the additional layer concept, which enables interoperability by having a layer with transformation logic on top of different Grid technologies (i.e. Grid middleware). This transformation logic is responsible to change the job description formats and protocols to the corresponding ones supported by the respective middleware. This concept is implemented in Grid portals like GridSphere [12] or APIs like JavaGAT [13] or GPE [14] and thus this additional layer is often located on the client-side.

The fundamental idea of the bridge approach is to introduce a neutral protocol that can be always used by clients since it is not affected to changes in the Grid middlewares. This neutral protocol is used to contact the neutral bridge implementation, which in turn uses its transformation logic to change the neutral protocol in the different proprietary formats for each of the corresponding Grid middlewares. This approach is taken to achieve the interoperability between the CORBA-based Integrade middleware and Globus Toolkits as described by Stone et al. in [15].

The gateway approach refers to one central entity that is able to translates any middleware protocol into any other middleware protocol using its transformation logic. It is used, for instance, to realize the interoperability between the European infrastructure EGEE and VEGA, which is the Grid Operating System (GOS) for the CNGrid infrastructure in China. Kryza et al. describes in [17] that the interoperability is achieved via a universal interoperability layer named as Grid Abstraction Layer (GAL) that can be seen as one instance of a gateway. By implementing the gateway approach, the GAL not only enables the interoperability between egee and vega, but also allows for the integration of any other Grid environments.

The mediator approach is similiar to the neutral bridge approach, but instead of using a neutral protocol the respective client technology sticks to one specific protocol A. This protocol can be used to access all Grid middleware's that natively support this protocol A, but also it can be used to access known mediators. These central mediators are always used via one specific protocol, but are in turn able to translate it into any other protocols with their implied transformation logic. This approach is adopted in the technologies that make EGEE interoperable with BOINC-based infrastructures as described by Kazsuk et al. in [16].

Another often applied approach is the adapter approach. This means a typical Grid middleware client submits with Protocol A its job to the respective Grid middleware, which in turn, after processing the job description, executes the job or forwards it to a dedicated adapter. This adapter in turn provides the transformation logic that transforms the job into the format of the corresponding other Grid middleware. Hence, the difference to other approaches such as mediator is that the Grid job is actually processed in one middleware stack before being forwarded to another middleware stack B for execution. This approach is adopted to achieve the interoperability between UNICORE 5 and gLite as described by Riedel et al. in [2].

6 Conclusion

In this paper, we raised the demand for an infrastructure interoperability reference model to promote interoperability between production Grids today. We have shown the basic design reference model and have highlights some of their core building blocks in the context of computation.

Since our work is fundamentally based on lessons learned from real production Grid interoperability use cases, we using improvements of common open standards, which in turn are already deployed on the production infrastructure. Hence, we many of the core building blocks of the IIRM and many of them are already deployed on the infrastructures and only minor changes (i.e. missing links, refinements, etc.) have to be done in order to achieve interoperability in production Grid infrastructures today.

Since our evaluation use cases have been very successful, we have given the IIRM as an input to OGF by creating a GIN spin-off activity named as the PGI working group. By chairing this group, our goal is to standardize the IIRM elements and thus feed back our valuable production experience into the standardization process of OGF.

With having participants from many important Grid infrastructures such as DEISA, EGEE, NGS, NorduGrid, and ARC, we are looking forward to get the core building blocks for our proposed IIRM design standardized very soon. This will significantly contribute to the vision of having an interoperable united federation of world-wide Grid infrastructures in the near future offering standardized access.

References:

[1] Riedel, M., et al.: Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA. In: Proceedings of the MIPRO (2007)

[2] Riedel, M., Laure, E., et al.: Interoperation of World-Wide Production e-Science Infrastructures. In: Journal on Concurrency and Computation: Practice and Experience (2008)

[3] M. Riedel, F. Wolf, D. Kranzlmüller, A. Streit, T. Lippert *Research Advances by using Interoperable e-Science Infrastructures - The Infrastructure Interoperability Reference Model applied in e-Science*, Journal of Cluster Computing, Special Issue Recent Advances in e-Science

[4] Sim, A., et al.: *The Storage Resource Manager Interface Specification Version 2.2*. OGF Grid Final Document Nr. 129 (2008)

[5] M. Antonioletti, M. Atkinson, A. Krause, S. Laws, S. Malaika, N. Paton, D. Pearson, G. Riccardi , Web Services Data Access and Integration - The Core (WS-DAI) Specification, Version 1.0

[6] I. Foster *et al.*, *OGSA Basic Execution Service Version 1.0*. Open Grid Forum Grid Final Document Nr. 108, 2007.

[7] A. Anjomshoaa *et al.*, *Job Submission Description Language (JSDL) Specification, Version 1.0*. Open Grid Forum Grid Final Document Nr. 136, 2008.

[8] Website, "GLUE2 Open Grid Forum Grid Forge Page,"https://forge.gridforum.org/sf/projects/glue-wg.

[9] Cantor, S., Kemp, J., Philpott, R., Maler, E.: Assertions and Protocols for the OASIS Security Assertion Markup Language. OASIS Standard (2005).
http://docs.oasisopen. org/security/saml/v2.0/

[10] Riedel, M., et al.: *Classification of Different Approaches for e-Science Applications in Next Generation Computing Infrastructures*. In: Proceedings of the Int. e-Science Conference, Indianapolis, USA (2008)

[12] "GridSphere," http://www.gridsphere.org/.

[13] V. Rob *et al.*, "User-friendly and Reliable Grid Computing Based on Imperfect Middleware," in *Proceedings of the International Supercomputing Conference 200, Reno, USA*, 2007.

[14] R. Ratering *et al.*, "GridBeans: Supporting e-Science and Grid Applications," in *2nd IEEE International Conference on e-Science and Grid Computing (E-Science 2006), Amsterdam,The Netherlands*, 2006.

[15] D. Stone *et al.*, "A Model for Transparent Grid Interoperability," in *Proc. of the CCGrid 2007 Conference*, 2007.

[16] P. Kacsuk *et al.*, "Towards making BOINC and EGEE Interoperable," in *Proc. of the IGGIW Workshop, e-Science Conference 2008, Indianapolis, USA*, 2008.

[17] Kryza, B., Skital, L., Kitowski, J., Li, M., Itagaki, T.: Analysis of Interoperability Issues Between EGEE and VEGA Grid Infrastructures. Springer-Verlag (2006)

[18] M. Riedel and D. Mallmann, "Standardization Processes of the UNICORE Grid System," in *Proceedings of 1st Austrian Grid Symposium 2005, Schloss Hagenberg, Austria*. Austrian Computer Society, 2005, pp. 191–203.

[19] EUFORIA project, http:// www.euforia-project.eu/

[20] S. Holl, M. Riedel, *et al.*, "Life Science Application Support in an Interoperable E-Science Environment," in *Proceedings of IEEE CBMS 2009, special Track: Healthgrid Computing - Applications to Biomedical Research and Healthcare*, 2009.