GES 2007 German e-Science

Available online at http://www.ges2007.de

Requirements and Design of a Collaborative Online Visualization and Steering Framework for Grid and e-Science infrastructures

Morris Riedel[1], Wolfgang Frings[1], Sonja Dominiczak[1], Thomas Eickermann[1], Thomas Düssel[1], Paul Gibbon[1], Daniel Mallmann[1], Felix Wolf[1], Wolfram Schiffmann[2]

[1] John von Neumann Institute for Computing
Central Institute for Applied Mathematics
Forschungszentrum Jülich GmbH, 52425, Jülich, Germany
[2] Institute of Computer Architecture, Department of Computer Science
University of Hagen, 58097 ,Hagen, Germany
email: {m.riedel}@fz-juelich.de
phone: (+49 2461) 61 3651, *fax:* (+49 2461) 61 6656

Abstract

Many production e-Science infrastructures (e.g. DEISA, D-Grid) have begun to offer a wide variety of services for end-users during the past several years. Many e-Scientists solve their scientific problems by using parallel computing applications on clusters and collaborative online visualization and steering (COVS) is known as a tool for analyzing and better understanding of these applications. In absence of a widely accepted COVS framework within Grids, visualizations are often created using proprietary technologies assuming a dedicated scenario. This makes it feasible to analyze the usual requirements to provide a blueprint for a more general COVS framework that can be integrated into Grid middleware systems such as UNICORE, gLite, or Globus Toolkits. These requirements lead to a design that was successfully implemented as a higher-level service in UNICORE and presented at numerous places such as the Open Grid Forum 19 and 20, Europar 2006, Supercomputing 2006 and DEISA trainings.

1 Introduction

Modern large-scale scientific research often relies on the collaborative use of a Grid or e-Science infrastructure (e.g. DEISA, EGEE) with computational or other types of physical resources. One of the major goals of these infrastructures is to facilitate the routine interaction of scientists and their work with advanced problem solving tools. Many e-Science applications within these Grids aim at numerical simulations of physical, chemical, biological processes or other scientific domain-specific problems and *parallel computing* is widely accepted as an essential tool for the solution of these problems. In the context of parallel computing, visualization and steering is known as a tool for analyzing and better understanding of shared parallel applications that run on a cluster or supercomputer. The concept to visualize complex scientific datasets (e.g. vectors, arrays)

lead often to more insights in the computational process of the application. Furthermore, a wider range of control is given through steering of the application by influencing its parameters during runtime. Non photo-realistic visualizations are used, since the rather schematic visual representations are often able to convey more information augmented with specific details such as scales or numbers that improve the insights of scientists.

At the time of writing, the major Grid middleware systems that provide access to computational resources offer no interactive services for schematic *collaborative online visualization and steering (COVS)* of parallel simulations, even if this technique is well established among scientists. This motivates this work by identifying the necessary functionality and quality of such a higher-level Grid service. Such service enables scientists to observe the intermediate steps during the computation of the simulations (*online visualization*) and can interact with the parallel simulation at once to influence its computation (*computational steering*). The integration of these services into Grids leads to several benefits for end-users, including single sign-on (1 times password provisioning) and virtualization of resources (no knowledge of hostname and username details). In absence of a widely accepted framework for COVS services, the most visualizations are often created by some proprietary mechanisms.

The lack of a common framework for COVS and standardized methods to create an online connection between a simulation and visualization is contrary to fundamental design principles of software engineering. This paper identifies requirements for a proposed design of a COVS framework within e-Science infrastructures. This includes the addressing of challenges in the area of visualization, communication, collaboration, as well as network and Grid issues. This requirement analysis lays the foundation for a general design of a COVS framework and its reference implementation using UNICORE [3] and VISIT [7].

The remainder of this paper is structured as follows. In Section 2 we define several requirements for a COVS framework in Grids. Section 3 presents the general design of the COVS framework, while Section 4 describes its implementation within UNICORE Grids and use case scenarios. The paper ends with a survey of related work and some concluding remarks.

2 Requirements for a COVS Framework in Grids

The integration of a COVS framework into e-Science infrastructures implies that specific requirements of these environments must be addressed. In addition, a COVS framework must satisfy all specific requirements for a scientific visualization and steering tool. The addressing of all these requirements lay the foundation for a COVS framework that allows for *scientific visualization*, defined by R.B. Haber et al. in [12]. In this sense, scientific visualization is the use of computer imaging technology as a tool for understanding, analyzing, and illustrate complex and comprehending data typically obtained by parallel simulations. Lessons learned from e-Science in the context of COVS indicate that the demands for visualization are increasing with the number of new applications in Grids and often conflicting requirements for a COVS framework.

2.1 Challenges in Visualization and Communication

A COVS framework should support a wide variety of existing visualization technologies (e.g. VTK, AVS/Express) in order to circumvent duplicate efforts in re-developing them and to gain the experience from the visualization community. The integration of such systems into a COVS framework provide the functionality to create *visualization idioms* defined by R.B. Haber et al. in [12]. A visualization idiom is a specific sequence of data enrichment and enhancement transformations, visualization mappings, and rendering transformations that produce a schematic display of a scientific dataset.

Another requirement is the support of communication libraries for visualization and steering (e.g. VISIT [7], PV3 [13]) that are used to transport datasets between the visualization and simulation and convey additional information such as steering commands. Hence, these powerful technologies decouple the visualization from the simulation and also decouple both from communication issues. Another fundamental challenge for a COVS framework is to allow for online visualization. This implies that the framework must transfer scientific data between the simulation and visualization in a stepwise fashion to allow for online visualization of single computational steps and thus for computational steering. The demand to use parallel computational resources as effective as possible leads to the requirement of adding computational steering technologies (e.g. VISIT, gViz [1], ICENI [16]) into the COVS framework. This enable scientists to focus on special areas of computation or to early abort computations that turn out to be false during its visualization. Finally, a COVS framework requires a secure bi-directional data transfer between the visualization and simulation with high performance, because scientific data computed by the simulation flows from the simulation to the visualization, while the steering data defined within the visualization flows from the visualization to the simulation.

In this context it seems reasonable to take the underlying network infrastructure into account that should provide an adequate bandwidth to support fast connections that in turn realize real-time behaviour of online visualizations. DEISA and TeraGrid, for instance, provide perfect bandwidth capabilities due to dedicated connections. Also, to use steering as effective as possible, a COVS framework should provide protocols with minimal overhead in order to use bi-directional connections with low latencies during COVS sessions.

2.2 Requirements for Collaborative Sessions

This section emphasize on the collaborative nature of COVS sessions with geographically dispersed participants that leads to the distinction of separate roles. First and foremost, a person that use the COVS framework is in the *participant role* if the person shares the view on one visualization of a parallel simulation with all other n-1 participants. While some people only act in the participant role, there are other people that may represent more than one role. This implies that the *functionality of the COVS framework for one role differs from the functionality offered to other roles.* For instance, a person that use the COVS framework is the *master* of the COVS session if this person uses the framework

to submit and control a parallel simulation that runs on a computational Grid resource. Hence, other participants do not need to submit a simulation job. A person in the *approver role* uses the COVS framework and makes decisions which participants are allowed to join COVS session. Thus, a person that uses the COVS framework to apply for participation in a COVS session is named a *candidate* and a person in this role needs approval by a person in the approver role to become a participant of a COVS session. From this it follows that some candidates may not get an approval since they are not allowed to share the view in the COVS session. Also, the master role is able to explicitly exclude participants from the session, if their behavior or technical reasons require such rather aggressive actions. In both cases the person acts in the *blocked participant role*.

Furthermore, capabilities to steer a parallel simulation during a collaborative session raises the requirement for a mechanism of mutual exclusion of participants during steering. Hence, only one participant in the *steerer role* is allowed at the same time to steer a parallel simulation during a COVS session in order to ensure the consistency of the simulation and its computation. In addition, only participants in the *collaborator role* are allowed to change the view of the visualization, which is not necessarily steering of the simulation itself, e.g. in case of turning the viewpoint of already computed data by 45 degrees.

Also, other challenges include the distribution of data and control information. A COVS framework requires a *multiplexer entity* that multiplexes the output of 1 parallel simulation to n bi-directional connections that provide n online visualizations of participants with the same data. This in turn raise a demand for *scalable multiplexing* in the sense that n participants can still share the same view of the data without an appreciable loss in quality of the scientific visualization. Second, a COVS framework requires an *collaboration entity* that transports the collaboration data (e.g. change level of detail, colors) from 1 visualization to all the other n-1 visualizations to ensure that all participants in the COVS session share the same view on the data. Thus, if one visualization makes a turn of 180 degrees, all other visualizations should also turn 180 degrees.

Finally, several requirements are related to the scope of session control. The *dynamics of collaborations* among the scientists lead to the demand of dynamically attaching and detaching participants to the visualization during the runtime of the simulation without influencing the quality of visualizations from other participants. This in turn raises a demand to monitor the *status of participants*. Thus, in order to determine whether all participants of a COVS session are already connected or still connected to the parallel simulation, the COVS framework requires a monitor mechanism that provides status information about the connection of all participants. This monitor mechanism should also include performance statistics of the bi-directional connections to all n visualizations of the participants in order to detect those that represent *bottlenecks* of a collaborative session and thus influence the overall quality of the session.

Finally, the overall management of a COVS session requires *authorized session management control actions* within the COVS framework that include the addition or removal of participants that may represent bottlenecks.

3 Design and Implementation of the COVS Framework

The design presented here provides an *architectural blueprint of a COVS framework* that can be implemented in different Grid environments that typically are based on different Grid middleware systems (e.g. UNICORE, gLite, or Globus Toolkits). One of the key considerations of this framework is to define an architecture that works for *a wide variety of applications in the scientific domain*. The design requirements identified in the previous section are most crucial to the framework design. A COVS framework that addresses them is more likey to achieve high levels of design, component and thus code reuse by still providing much flexibility. The general design and the implementation presented here satisfy all the described requirements by using UNICORE as a Grid middleware and VISIT as the communication library. This section highlights several crucial framework parts in more detail while an overview of the COVS framework design is illustrated in Figure 1.

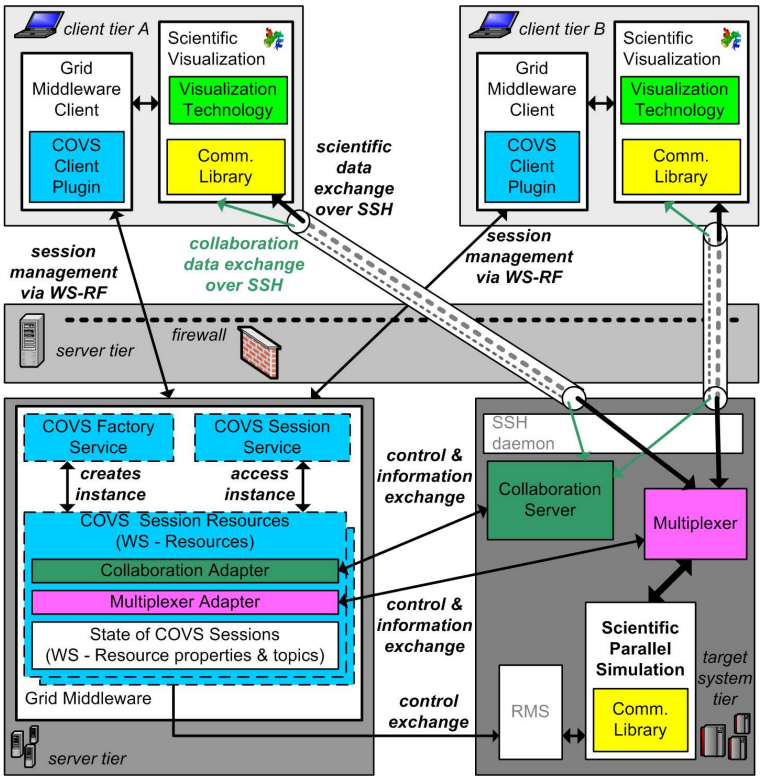

Figure 1: COVS architecture for UNICORE, gLite or Globus Toolkits as Grid middleware. Web services are used for communication as well as SSH tunnels for the bi-directional transfer of scientific data and steering commands.

An adequate framework for COVS that is built on Grid computing and communication technologies relies on technologies provides by the Grid and e-Science infrastructure. In this context, the COVS framework is based on *Grid middleware* systems to gain all the benefits of these systems and thus of the whole infrastructure. Hence, the COVS framework is integrated seamlessly into the Grid or e-Science infrastructure by the meaning of hiding the fact that resources are physically distributed and tpically managed via *Resource Management Systems (RMS)* such as Torque, LSF or LoadLeveler. Such a *transparency* includes differences in security policies, data representation and how a resource is accessed when using the Grid services of the COVS framework.

The Grid middleware represents a crucial core building block for *simulation job management* as shown in Figure 1. It is capable of submitting, controlling and managing computational jobs such as a parallel simulation that run on supercomputers or clusters. This is mainly supported by core Grid services such as the *UNICORE Atomic Services (UAS)* [14] or *OGSA - Basic Execution Services (BES)* [4]. However, the Grid middleware must also be *extensible* to allow for the creation of additional Grid services that represent higher-level services for COVS sessions. In particular, the design of the framework relies on the *Web Services Resource Framework (WS-RF)* [9] standard for the development of such services. In more detail, the WS-RF compliant *COVS Grid service* represents another core building block and is used for COVS session management. The *COVS Factory service* implements the WS-RF factory pattern [9] and brings *COVS Session Resources* with properties into existence that are in turn accessed by the *COVS Session service*. As shown in Figure 1, the COVS Session service interacts with the *multiplexer* and *collaboration server* and exposes the status of the COVS session as WS-Resource properties [9]. This allows the COVS Grid service to have the explicit control over the scientific dataflow to numerous participants and the collaboration data exchange. To sum up, the COVS framework relies on middleware that offers services via *open connection technologies* according to standard rules or emerging Grid standards such as WS-RF. We implemented the COVS Grid service within the WS-RF based UNICORE 6 Grid system [11].

The integration of COVS functionality into Grids implies that the fundamental *authorization and authentication* of end-users is managed by the corresponding Grid middleware. In the context of UNICORE, only end-users that are configured within the *UNICORE user database* [3] are able to participate in a COVS session. Of course, the seamless integration into the security infrastructure in Grids places the requirement on the COVS framework to *preserve single sign-on*. In more detail, the design of the COVS framework relies on SSH for the bi-directional data transfer and thus a typical end-user must know a concrete hostname, username of the remote host to establish the connection. But to preserve the single sign-on the Grid middleware provides all these necessary details and thus retains transparency to end-users. As shown in Figure 1, the Grid middleware client and the scientific visualization interact with each other. This interaction is used to transfer all necessary details to establish an SSH connection from the scientific visualization to the parallel simulation. In the context

of the implementation with UNICORE, we use an *RSA-based authentication*. A SSH session key is generated in the client and transferred to the target system via UNICORE. Next, the public key is inserted into the *authorizedkeys* file of the target system and thus the client tier gets access per SSH. When the visualization session is finished, the key is removed. This approach is similar to the SSH interactive access to UNICORE as described by Riedel et al. in [5].

E-scientists use Grid resources with parallel computing techniques to solve problems in their areas of science. Today, most parallel computers of supercomputing centers are totally booked out or their demand is five times higher than what the resources can offer. Thus, computational time on parallel supercomputers is not cheap and *should be used as effectively as possible*. In this context, the COVS framework must support a minimization of the load on a steered parallel simulation and the circumvention of failures or slow operations by the visualization that may disturb the simulation progress. We achieved that by using the VISIT communication library. In VISIT, the *simulation acts as a client which initiates all operations* like opening a connection through the established SSH tunnel, sending data to be visualized or receiving new steering parameters.

Also, we identified a demand for a managing *Grid client* (e.g. GPE Client suite [8], or CogKits) within the COVS framework that provides an overview of the session by using information exposed by the underlying Grid middleware.

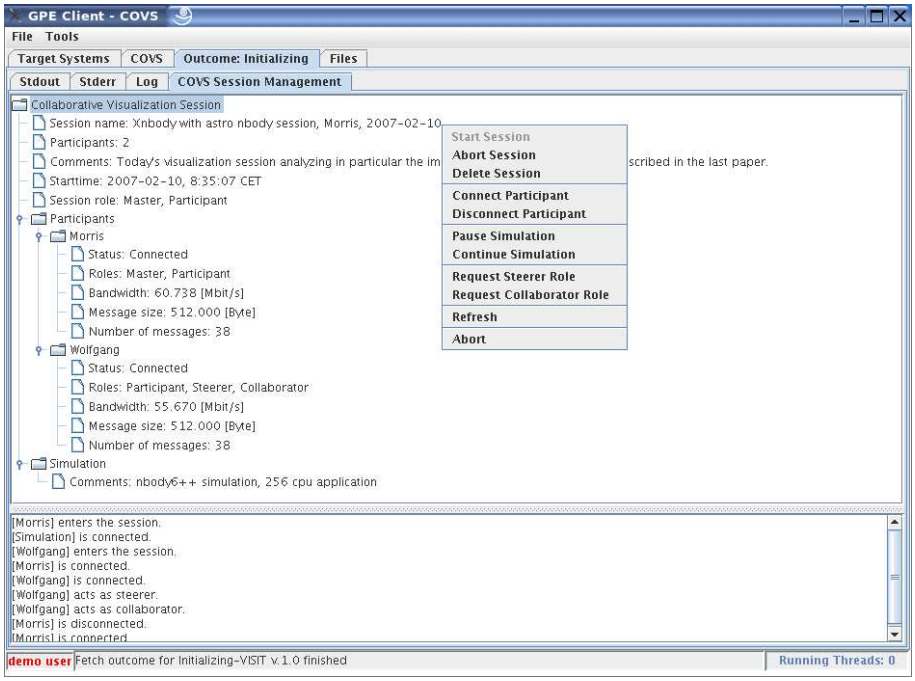

Figure 2: COVS GridBean for collaborative session management. End-users can use popup menus to conveniently monitor and manage COVS sessions.

Therefore, the Grid client as well as its *COVS specific plugin* for COVS session management also represent core building blocks. In our implementation, we used the GPE application client and developed a COVS GridBean as shown in Figure 2. GridBeans are scientific application-specific plugins for the GPE clients [8]. Finally, the *parallel simulation* and *scientific visualization* represent core building blocks that are scientific application-specific and thus it is feasible provide an concrete example in the next Section.

4 Use Case Scenarios

In order to demonstrate the re-usability for parallel simulations and scientific visualizations, we provide a concrete use case scenario that is based on the *scientific visualization Xnbody* [15]. It is based on VTK and integrates the VISIT toolkit as shown in Figure 3. Xnbody shows the output of parallel simulations in the context of n-body problems by using the COVS framework implementation in UNICORE. In particular Xnbody is used in the context of plasma physics (PEPC simulation code) and astrophysics (nbody6++ code). Without using the COVS framework implementation end-users have to manually provide details about the SSH connections to remote sites. When using the COVS framework implementation *UNICORE provides all necessary details for the SSH connection* and thus provides transparency and single sign-on to end-users of Xnbody, including the session management capabilities of the COVS GridBean.

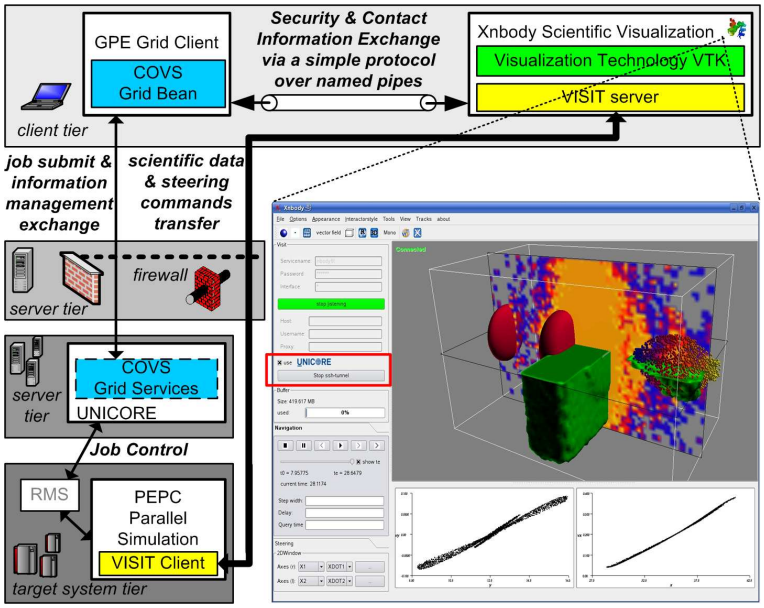

Figure 3: Xnbody uses the 'use UNICORE' checkbox to seamlessly connect to supercomputing sites via security information based on UNICORE.

5 Related Work

There is a wide variety of related work in the field of visualization and steering within Grids. For instance, Brodlie et al. describe in [10] a framework for distributed and collaborative visualization and how it can be potentially implemented by several visualization systems. But the framework is rather high-level and thus we followed an approach that was closer to current production Grids. Furthermore, there is considerable research interest in new visualization and steering technologies within many national and international Grid initiatives.

First and foremost, Kleijer et al. describe in [2] the API for the Grid-based visualization systems of the NAREGI Grid infrastructure. This API consists of a visualization library and a Grid visualization service API that provides Grid service functionality conform to the Open Grid Services Architecture (OGSA). In the last years, the API evolved to a wide-variety of WS-RF compliant services for visualization such as a post-processing service. Even if WS-RF is also used in our approach, we use a different approach since the NAREGI services are rather image-based instead of following the online visualization technique. That means that the scientific data as well as their rendering and visualization are completely computed within the Grid that finally leads to a compressed image that is transferred to clients and thus makes it difficult to apply steering.

Köckerbauer et al. describe in [6] the visualization system that is used in the context of the Austrian Grid. This system is named as Grid Enabled Visualization Pipeline (GVID) and provides high quality Grid-based visualization of scientific datasets on thin clients (e.g. playstation clients). In more detail, the scientific data is efficiently encoded with the H262 code into a video stream and transferred to the client afterwards. The client in turn decodes the video stream for visualization and can apply steering commands that are tracked by the GVID Event-Encoder. Finally, GVID is used for Galaxy visualizations, but the major difference to our approach is that it is not seamlessly integrated as a higher-level service into a specific Grid middleware.

The UK RealityGrid project focused on how scientists can make more effective use of a Grid and its visualization resources. The most known work in RealityGrid is around its steering library that enable calls which can be embedded into each of the three components of its architecture, namely simulation, visualization, and a dedicated steering client. More recently, older prototypes of RealityGrid are renewed towards OGSA environments and unfortunately tightly integrated into the Imperial College e-Science Networked Infrastructure (ICENI) [16]. Inversely, the COVS framework is rather loosely coupled from the underlying Grid middleware.

6 Conclusions

The design of the COVS framework provides a sophisticating blueprint for implementations in different Grid middleware systems. Thus, to demonstrate that the COVS requirements and problems addressed within this paper are of

practical relevance, we implemented a COVS framework implementation in UNICORE by using the VISIT toolkit. *This implementation is ready for production and can be used by all VISIT-enabled simulations and visualizations.* The seamless integration of the COVS framework into UNICORE Grids improves the work of scientists by providing online schematic visualizations of complex parallel simulations by still retaining the single sign-on feature of Grid and e-Science infrastructures and secure data transfers with SSH across Grids.

Finally, the work within this paper was successfully demonstrated at the OGF18, Europar 2006, Supercomputing 2006, and recently at the visualization workshop at OGF19. Furthermore, it was demonstrated to end-users in DEISA at the DEISA training in November 2006 and is continously shown as one example of an higher-level service in UNICORE at various UNICORE presentations world-wide. Nevertheless, deploying the proposed COVS architecture is an important next step to broadly incorporate implementations of a COVS framework into production Grid environments. Once an implementation of the COVS framework is deployed within production Grids such as DEISA or D-Grid, an important tool for an efficient use of the Grid is accomplished.

References

1. K.Brodlie et al. Visualization in Grid computing environments: Technologies, Applications and Challenges. In Computer Graphics Forum, Vol.24(2), 2005
2. P. Kleijer et al. API for Grid Based Visualization Systems. In GGF 12 Workshop on Grid Application Programming Interfaces, 2004
3. A. Streit et al. UNICORE - From Project Results to Production Grids, Elsevier, Grid Comp. and New Frontiers of High Perf. Proc., pages 357–376, 2005
4. OGSA - BES, http://www.ogf.org/gf/group_info/viewṗhp?group=ogsa-bes-wg.
5. M. Riedel et al. Enhancing Scientific Workflows with Secure Shell Functionality in UNICORE Grids, In Int. e-Science conference, Melbourne, 2005
6. T. Köckerbauer et al. GVid - Video Coding and Encryption for Advanced Grid Visualization. In Proc. of the 1st Austrian Grid Symposium, Linz, 2005
7. Visualization Interface Toolkit, http://www.fz-juelich.de/zam/visit
8. R. Ratering et al. GridBeans: Supporting e-Science and Grid Applications. In 2nd Int. e-Science conference, Amsterdam, 2006,
9. OASIS - WS-RF Technical Committee, http://www.oasis-open.org/committees/tc_home.php?/wg_abbrev=wsrf
10. K.Brodlie et al. Distributed and Collaborative Visualization. In Computer Graphics Forum, Vol.23, 2004
11. UNICORE Grid Middleware, http://www.unicore.eu
12. R.B.Haber and D.A.McNabb. Visualization idioms: A conceptual model for scientific visualization systems, In Vis. in Scientific Comp., pages 74-93
13. PV3, http://www.raphael.mit.edu/pv3
14. M. Riedel et al. Standardization Processes of the UNICORE Grid System. In Proc. of 1st Austrian Grid Symposium 2005, Linz, Austria, pages 191-203
15. Xnbody - Visualization, http://www.fz-juelich.de/zam/xnbody
16. J. Cohen et al. RealityGrid: An Integrated Approach to middleware through ICENI. In Phil. Transactions of the Royal Society, 363, pages 1817-1827, 2005