Problem solving environment for distributed interactive applications.

Katarzyna Rycerz[1,2]
Marian Bubak[1,2]
Peter M.A. Sloot[3]
Vladimir Getov[4]

[1] Institute of Computer Science AGH, Krakow, Poland
[2] Academic Computer Centre – CYFRONET, Krakow, Poland
[3] Faculty of Sciences, Section Computational Science, University of Amsterdam
[4] Harrow School of Computer Science, University of Westminster

This is a reproduction of CoreGRID Technical Report Number TR-0107, 30 August 2007, and is reprinted here with permission.

The report is available on the CoreGRID website, at:

http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0107.pdf

---

European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Problem Solving Environment for Distributed Interactive Applications

Katarzyna Rycerz[1,2], *Marian Bubak*[1,2], *Peter M.A. Sloot*[3], *Vladimir Getov*[4]
{kzajac,bubak}@agh.edu.pl
{sloot}@science.uva.nl
{V.S.Getov}@wmin.ac.uk
[1]*Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland*
[2]*Academic Computer Centre – CYFRONET, Nawojki 11, 30-950 Kraków, Poland*
[3]*Faculty of Sciences, Section Computational Science, University of Amsterdam Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*
[4]*School of Computer Science University of Westminster Watford Rd, Northwick Park Harrow HA1 3TP, U.K.*

CoreGRID Technical Report
Number TR-0107
August 30, 2007

Institute on Grid Systems, Tools, and Environments

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Problem Solving Environment for Distributed Interactive Applications

Katarzyna Rycerz[1,2], Marian Bubak[1,2], Peter M.A. Sloot[3], Vladimir Getov[4]
{kzajac,bubak}@agh.edu.pl
{sloot}@science.uva.nl
{V.S.Getov}@wmin.ac.uk
[1]Institute of Computer Science AGH, al. Mickiewicza 30, 30-059
Kraków, Poland
[2]Academic Computer Centre – CYFRONET, Nawojki 11,
30-950 Kraków, Poland
[3]Faculty of Sciences, Section Computational Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[4]School of Computer Science University of Westminster
Watford Rd, Northwick Park Harrow HA1 3TP, U.K.

CoreGRID TR-0107

August 30, 2007

Abstract

Interactive Problem Solving Environments (PSEs) offer an integrated approach for constructing and running complex systems, such as distributed simulation systems. To achieve efficient execution of High Level Architecture (HLA)-based distributed interactive simulations on the Grid, we introduce a PSE called Grid HLA Management System (G-HLAM) for their management. This is done by introducing migration and monitoring mechanisms for such applications. In this paper we present how G-HLAM can be applied to the applications supporting surgeons with simulations of vascular reconstruction, using distributed federations on the Grid for the communication among simulation and visualization components.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

1 Introduction

Problem Solving Environments (PSEs) are integrated computational systems that allow scientists to define complex problems, find the required nearest components and resources available, and utilize them efficiently. PSEs offer an integrated approach for constructing and running complex systems, such as distributed simulation and decision support systems.

In this paper we focus on PSEs for running distributed interactive simulations on the Grid. This effort gives a potential opportunity for better and more convenient usage of distributed resources that are needed by such simulations, but were previously inaccessible and are now available through Grid.

There are solutions that may be used as underlying frameworks for such PSEs. One of them is the High Level Architecture (HLA) [11] which offers many features for developers of interactive and distributed applications. HLA enables merging geographically distributed parts (called *federates*) of simulations (called *federations*) into a coherent entity. It is explicitly designed as support for interactive distributed simulations, it provides various services required for that specific purpose, such as time management, useful for time-driven or event-driven interactive simulations. It also provides data distribution management and enables all application components to access the entire application data space in an efficient way. On the other hand, the HLA standard does not provide automatic setup of HLA distributed applications and there is no mechanism for migrating federates according to the dynamic changes of host loads or failures, which is essential for Grid applications. Therefore, there is a need for a PSE that would manage HLA-based collaborative environments on the Grid.

The Grid Services concept provides a good starting point for building the Grid HLA Management System (G-HLAM) for that purpose, as described in [23]. The concept of G-HLAM can be also ported to component platforms like CCA [4], H2O [14], ProActive [20] or Grid Component Model [10].

The paper is organized as follows: in Section 2 we present an overview of most important PSEs for distributed interactive simulations. For each of these environments, we analyse the advantages and disadvantages for adapting Grid solutions. In Section 3 we describe benefits of using HLA for our purposes and present G-HLAM system. In Section 4 experimental results are presented and we conclude in Section 5.

2 PSEs for distributed interactive applications

This Section presents the overview of Problem Solving Environments which may be applied for interactive distributed applications.

2.1 Computational Steering Environment

The aim of the Computational Steering Environment (CSE) [6] is to provide scientific end users with an environment in which they can easily define interactive interfaces to ongoing simulations. The CSE architecture is implemented as a set of processes - called satellites - which implement standard visualization operations. The simulation is also seen by the system as a satellite. Satellites cooperate by sending and receiving data from a central data manager which, in turn, notifies all interested satellites about data mutations. The most predominant satellite is the interactive graphics editing tool called Parametrized Graphics Object (PGO) editor, which allows the end user to sketch out visualizations. A two-way binding between visualization and data is achieved by binding the sketch to data within the data manager. CSE uses the TCP/IP protocol as a communication layer between satellites. The main disadvantage of the CSE is its centralization, which hampers its scalability in Grid environments. However, the idea of a data manager and satellites can be somehow extended (e.g. by building hierarchical or distributed data sets).

2.2 CUMULVS

Collaborative User Migration, User Library for Visualization and Steering
(CUMULVS) [13] allows the programmer to add interactive steering and visualization to an existing parallel or serial program (task). With CUMULVS, each of the collaborators can start up an independent view program that will connect to the running simulation program. Viewers allow scientists to browse through the various data fields being computed and observe the ongoing convergence toward a solution. CUMULVS also allows an application program to perform user-directed checkpointing and automated restarts of parallel programs using checkpointing, even across a heterogeneous cluster of machines. A single user library interface routine passes control to CUMULVS periodically,

to transparently handle the viewer attachment / detachment protocols, the selection and extraction of data, and the updating of steering parameters. CUMULVS allows each front-end viewer to interactively select the granularity and extent of data that it desires to view. Currently, CUMULVS uses PVM [21] as its message passing substrate; it allows for pairs of anonymous tasks to communicate with each other without both tasks being started at the same time. MPI does not allow these dynamics, so porting CUMULVS ideas to MPI would not be easy.

2.3 Cactus Problem Solving Environment

Cactus [1] is an open-source problem solving environment designed for scientists and engineers. The name Cactus comes from the design of a central core, which connects to application modules - or thorns - through an extensible interface. Thorns can implement custom-developed scientific or engineering applications, such as the Einstein solvers, or other applications such as computational fluid dynamics. Cactus is an environment for a wide range of issues, here we concentrate on its support for parallel or distributed simulations and their visualisation. In Cactus, different thorns can be used to implement different parallel paradigms, such as PVM, Pthreads [17], OpenMP [19], CORBA [5], MPICH-G etc. Cactus can be compiled with as many driver thorns as required (subject to availability), with the one actually used chosen by the user at runtime through a parameter file. Cactus provides the ability to stream online data from a running simulation via TCP/IP socket communications. Multiple visualization clients can connect to a running Cactus executable via a socket from any remote machine on the Grid, then request arbitrary data from the running simulation and display simulation results in real time. This can be done in two ways: by an HTTP control interface or through socket connections. Cactus already provides support for Grid–enabled MPI – MPICH-G. Its main disadvantage is that the parallelization is limited to domain decomposition. However, because of the modular architecture of Cactus, it appears that adding extended functionality would be quite easy for application developers.

2.4 Discover

Discover [15] is a interactive and collaborative system that enables geographically distributed scientists and engineers to collaboratively monitor, and control high performance parallel/distributed applications using web-based portals. Discover provides a three-tier architecture composed of detachable thin clients at the front-end, a peer-to-peer network of servers in the middle, and the Distributed Interactive Object Substrate (DIOS++) at the back-end. An important part of Discover is a rule-based visualization system. Rules are decoupled from the system and can be externally injected to manage such visualization behavior at runtime, as autonomically selecting the appropriate visualization routines and methods and adjusting the extraction threshold. To allow rule–based management, DIOS++ provides autonomic objects that extend application computational objects with sensors to monitor the state of the objects. It also contains actuators to modify the state of the objects, access policies to control accesses to sensors and actuators and rule agents to enable rule-based autonomic self-management. Discover uses HTTP for connection with visualisation thin clients and CORBA for communication with application engines. Discover already possesses a distributed and scalable peer-to-peer type of architecture. However, it can still be extended to take advantage from Grid technology, which would enable it to be automatically set up and effectively run in a non-centrally controlled environment consisting of different administrative domains.

2.5 TENT

TENT [24] is a software integration and workflow management system that helps improve the building and management of process chains for complex simulations in distributed environments. TENT allows for online steering, and visualization of simulations. Wrappers are used to interface application modules (e.g. Computation Fluid Dynamics (CFD), Computation Structural Mechanics (CSM), visualisation or filters) with the system. The system consists of base components which include: modules for controlling workflows; factories for starting system and applications in the distributed environment; the name server as the central information service. There are also support components – additional services for special application scenarios not covered by the basic functionality. Examples include: a data server for storing data files, a monitoring and reporting component, and several special control components (e.g., for coupled simulations like the ones parallelized with MPI). The system is controlled by a user through a GUI. TENT uses CORBA for communication between parts of the system. TENT is already Grid–enabled: it supports MPICH-G2 simulations. The Globus Toolkit version 2 is used for resource selection, for starting applications, and for data transfer and security.

2.6 Interactive Simulation Systems Conductor

Interactive Simulation Systems Conductor (ISSConductor) [27] is an agent oriented component framework for Interactive Simulation Systems. The system introduces two kinds of agents: Module Agents that are specific for different modules of interactive application like simulation, visualization and interaction and Communication Agents that are used for communication between Module Agents and actual modules. Module Agent uses an extended finite state machine to model the run-time behavior of a component, and adopts first order logic to represent the interaction constraints between components and to implement them in the knowledge bases of agents. ISS-Conductor separates the basic computational functions of a component from its run-time behavior controls, and provides a high level interface for users to design interaction scenarios. The framework is very general and can be used for various interactive applications. ISS-Conductor is built on top of HLA described in the previous Section. Currently, ISS-Conductor is not Grid-enabled, however it can easily take advantage of the system presented in this paper, since it is built over the HLA standard.

2.7 Comparision of existing PSE's

System	Type of simulation and its distribution	Protocol	Porting to the Grid issue
CSE	runtime steering – multiple visualizations for one simulation	TCP/IP	scalability issue
CUMULVS	runtime steering of parallel simulations (PVM)	PVM	porting to MPI issue
CACTUS	runtime steering of parallel or distributed simulations PVM, Pthreads, OpenMP, CORBA, MPICH-G	two possibilities: 1) HTTP 2) HDF5 format over TCP/IP	support for MPICH-G
Discover	parallel and distributed applications in general	HTTP, CORBA	scalable architecture that could be extended to allow automatic setup and effective run in non-centrally controlled Grid environment
TENT	parallel and distributed simulations	CORBA	TENT is already Grid enabled by using GTv2 and MPICH-G2 features [24]
ISS-Conductor	distributed simulations	HLA	no adaptation to changing environment, no automatic setup, no dynamic discovery

Table 1: Main features of the interactive simulation and visualization environments

In this Section we have presented environments that support interactive steering of simulations. For each of presented systems we described the architecture and protocols used to connect simulations with visualizations. For each of the systems we also analyzed the possibilities of adapting it to the Grid environment. A summary of the features of the presented systems is show in Tab. 1. CSE, CUMULVS, Cactus, Discover and TENT focus more on support

for steering simulations without much concern for advanced simulation composition from distributed components which often use different time management. ISS-Conductor is a high–level system built over HLA - a standard allowing for interoperability between different types of simulations. Basing on this analysis, we have chosen HLA as a base for distributed interactive simulations running on the Grid. It is a well recognized standard and offers all the necessary functionality for simulation developers. Its important feature is that the local time management mechanism of one simulation component (federate) is not visible to other federates. Hence, all forms of time management (time–driven, event–driven, parallel discrete event, real–time–driven) may be linked together. HLA also allows to build scalable simulation systems. It separates the communication infrastructure from the actual simulation. Additionally, it introduces a uniform way of describing events and objects being exchanged between federates. All of these features allow interoperability between various simulations. Although HLA originates from the defense technology, there is a growing interest from non-military areas like manufacturing, transportation and gaming industries. Therefore companies are currently working on more scalable and efficient implementations of the standard [22]. Recently, open source implementation was also released [18].

3 PSE for HLA-based simulations

3.1 Need for a Grid for HLA-based applications

Usually, parts of distributed simulations require different resources: quick access to database, computational power or specific VR hardware. It is quite unlikely to find those resources at one geographical site. Additionally, if more simulations need to be run concurrently, one site with computational power may not be sufficient. A similar problem arises when many visualisations (users) located in different places want to observe the same simulation. Therefore, the application modules usually have to be located in geographically different places and the Grid concept that facilitates access to computing resources may be a very promising approach here.

As stated above, HLA has advanced mechanisms supporting distributed simulations, so execution of HLA–based applications on the Grid should be natural extension of its usage. However, the HLA standard was developed assuming a certain quality of service in the underlying environment. Therefore, there is a need for adaptation of HLA–based applications to the dynamically changing Grid as well as for automatic setup, dynamic discovery and fault-tolerance mechanisms.

The Grid [7] is designed to coordinate resources that are not under central control. Additionally, the Web services [25] concept of abstract interfaces allows for modular design (OGSA, WSRF) [8]. However, the Grid environment is shared between many users and its conditions can change in an unpredictable way. Therefore, there is a need for a system that adapts HLA-based applications to a dynamically–changing environment and requires fault tolerance mechanisms such as migration of its distributed federates or their monitoring.

In addition, the Grid idea is to facilitate access to computing resources and make it more transparent to the user. Currently, setting up distributed applications based on HLA requires tedious setup and configuration. The HLA standard does not cover aspects of dynamic discovery of HLA federations. Therefore, there is a need for a mechanism that sets up an HLA–based application on geographically distributed system (i.e. the Grid) in a more convenient way. Subsequently, HLA federates should be able to find one another dynamically and transparently to the user. Additionally, HLA does not provide security mechanisms similar to the one provided by the Grid Security Infrastructure (GSI) [9],

3.2 Grid HLA Management System

The Grid HLA Management System (G-HLAM) supports efficient execution of HLA-based simulations on the Grid. The system is built on top of the Open Grid Services Infrastructure as presented in [26, 23, 12]. In the future we would like to use also lightweight Grid Component platform [2] for that purpose.

The architecture of G-HLAM system is shown in the Fig. 1. The group of main G-HLAM services consists of: a *Broker Service* which coordinates management of the simulation, a *Performance Decision Service* which decides when the performance of any of the federates is not satisfactory and therefore migration is required, and a *Registry Service* which stores information about the location of local services. On each Grid site, supporting HLA, there are local services for performing migration commands on behalf of the *Broker Service*, as well as for monitoring federates and benchmarking. *Application Monitoring Services* are based on the OCM-G monitoring system [3]. The *HLA–Speaking Service* is one of the local services interfacing federates with the G-HLAM system, using GRAM for submission of

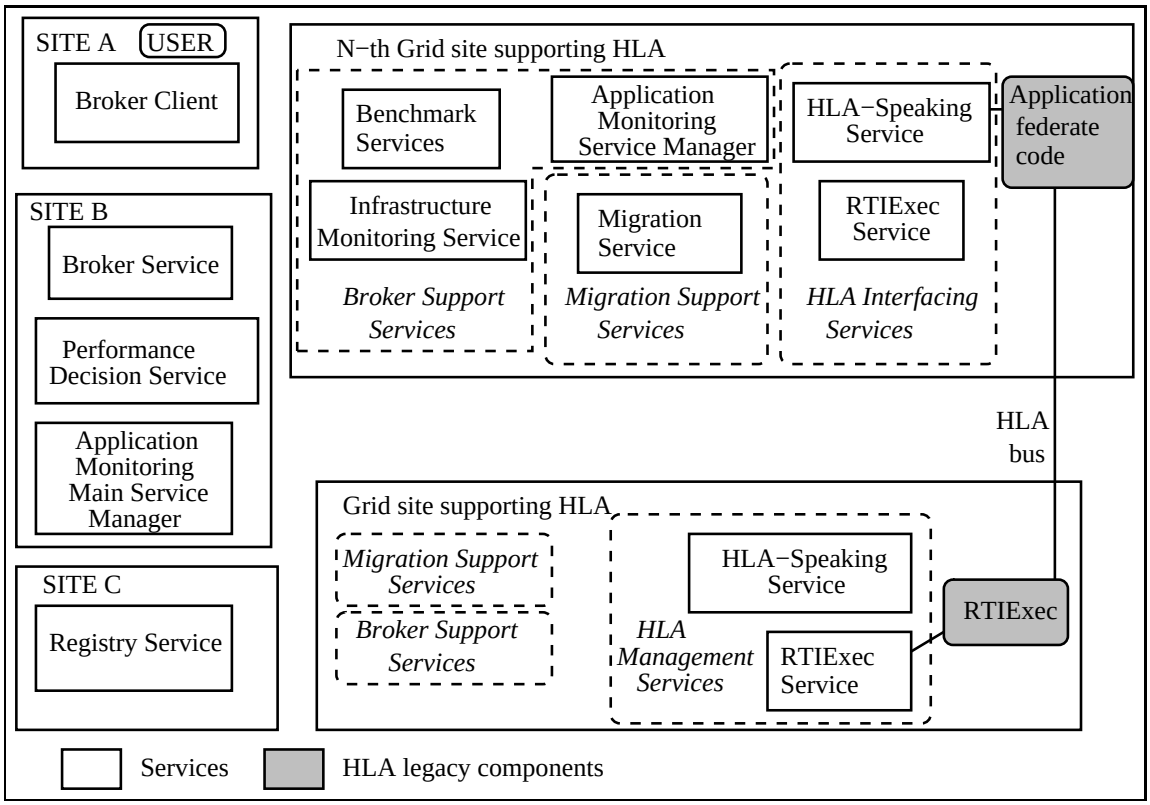

Figure 1: Grid HLA Management System Architecture (G-HLAM consists of services which control the whole HLA application and the services that should be installed on each HLA-enabled Grid site.

federates. A more detailed description of the *HLA-Speaking Service*, together with the GridHLAController library, which actually interfaces the application code with the system, can be found in [12].

4 Application of G-HLAM to vascular reconctruction

In this section we present results of the experiment in which G-HLAM was applied as the prototype collaborative environment for vascular reconstruction. The prototype consists of two types of modules communicating with HLA: *simulation module* (MPI parallel simulation) and *visualization-receiver modules* (responsible for receiving data from the simulation). At each time step, the simulation calculates velocity vectors of simulated blood flow in 3D space and sends them to visualisations modules. In our experiment, we have measured the duration time of first 8 steps of the simulation that included actual calculations and sending time.

We show how migration improves performance from the point of view of the user - i.e. how sending output data from the simulation changes after migration if the partial simulation results are actually observed by someone. The experiment was performed on the DutchGrid DAS2 testbed infrastructure and at CYFRONET, Krakow, as shown in Tab. 2. We used GT v3.2 and HLA RTI 1.3v5. In the presented experiment one simulation was migrated. The number of visualization-receivers was fixed and equal to 25. In this experiment we show how migration can improve the efficiency of simulation execution when its results are sent online to many users. The bandwidth available for testing was broad (10Gbps), so communication did not play an important role and calculations were the most time–consuming part of the execution. In order to create conditions in which migration would be useful, we increased the load of the Grid site where the simulation was executed (cluster in Amsterdam) by submitting non–related, computationally–intensive jobs. Next, we imitated a *Resource Broker* and migrated the simulation to another site which was not overloaded (cluster in Leiden). The actual migration was conducted by *Migration Service* using *HLA–Speaking Service*. The experiments were performed at night in order to avoid interference from other users. Tab. 3 shows duration times of interactive steps with a human in the loop (for the first 8 steps). At each step, the simulation calculates data and sends

Operating System	Red Hat Enterprise Linux Advanced Server, version 3		
Network	10 Gbps (DAS2) + 155 Mbps (DAS2-Cyfronet)		
Role	**Name**	**CPU**	**RAM**
Migration source	DAS2 Nikhef	Pentium III 1 GHz	1 GB
Migration destination	DAS2 Leiden	Pentium III 1 GHz	2 GB
visualizations	DAS2 Delft	Pentium III 2GHz	2 GB
	DAS2 Utrecht	Pentium III 1 GHz	1 GB
	DAS2 Vrije	Pentium III 1 GHz	2 GB
RTIexec	Cyf Krakow	Xeon 2.4 GHz	1 GB

Table 2: Grid testbed infrastructure

Number of simulation's step	**Calculations plus sending from sim. to vis. time**	**Note**
1	112	before migration
2	109	before migration
3	100	before migration
4	177	including migration
5	30	after migration
6	45	after migration
7	46	after migration
8	39	after migration

Table 3: Impact of migration on simulation performance within the collaborative environment

it to the 25 visualization-receivers modules using HLA. Tab. 3 shows that before migration the average time in a single step was around 107 sec and after migration around 40 sec (which is 2.6 times shorter). The time of the step when migration was performed was 1.6 longer then the average time before migration. The results show that it is better to spend some time on migration to another site, from where the response time is shorter. In our experiment, in each step, the simulation produces 52000 velocity vectors of simulated blood flow in 3D space.

5 Summary, Conclusions and Future Work

In this paper we have presented the brief analysis of PSEs supporting the development, execution and/or steering of simulations. For each of these environments, we have analysed the advantages and disadvantages for adaptation of Grid solutions. According to our analysis, there was no solution that allowed to run HLA simulations, including legacy codes, on the Grid in efficient way as it can be achieved by the system we have developed – G-HLAM. In particular, we have shown that migration of badly performing parts of simulations to a better location reduces computation and communication time and effectively improves the overall performance.

The future work will concentrate mainly on using component architectures (like CCA [4], H2O [14], ProActive [20], GCM [10]), as they are a very promising approach due to such features as lightweight environments [2], dynamic behavior, scalability to suit various environmental requirements etc. Applying the advantages of these technologies to distributed interactive simulations will allow not only for technological migration of existing G-HLAM functionality, but also for it's extension to achieve reusability and interoperability of simulation models.

Acknowledgments The authors would like to thank Piotr Nowakowski for useful remarks. This research is partly funded by the the Polish Foundation for Science (FNP), EU IST Project CoreGRID and the Polish State Committee for Scientific Research SPUB-M grant.

References

[1] G. Allen, W. Benger, T. Dramlitsch, T. Goodale, H. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, and J. Shalf. Cactus Tools for Grid Applications. *Cluster Computing*, 4(3):179–188, 2001.

[2] R. M. Badia, O. Beckmann, M. Bubak, D. Caromel, V. Getov, S. Isaiadis, V. Lazarov, M. Malawski, S. Panagiotidi, and J. Thiyagalingam. Lightweight grid platform: Design methodology. In S. Gorlatch and M. Danelutt, editors, *Proceedings of the CoreGRID Workshop "Integrated Research in Grid Computing, November 28-30, 2005*, pages 126–134, Pisa, 2005. Technical Report TR-05-22.

[3] B. Baliś, M. Bubak, W. Funika, T. Szepieniec, R. Wismüller, and M. Radecki. Monitoring Grid Applications with Grid-enabled OMIS Monitor. In F. Riviera, M. Bubak, A. Tato, and R. Doallo, editors, *Proc. First European Across Grids Conference*, volume 2970 of *Lecture Notes in Computer Science*, pages 230–239. Springer, Feb. 2003. `http://www.icsr.agh.edu.pl/ocmg`.

[4] The Common Component Architecture Forum, 2004. `http://www.cca-forum.org/`.

[5] CORBA project home page. `http://www.corba.org/`.

[6] CSE project home page. `http://www.cwi.nl/projects/cse/cse.html`.

[7] I. Foster. What is the Grid? A three checkpoints list. *GridToday Daily News And Information For The Global Grid Community*, 1(6), July 2002.

[8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002. `http://www.globus.org/alliance/publications/papers.php`.

[9] Globus project home page. `http://www.globus.org/`.

[10] GridComp project home page. `http://gridcomp.ercim.org/`.

[11] High Level Architecture specification. `http://www.sisostds.org/stdsdev/hla/`.

[12] K. Rycerz and M. Bubak and M. Malawski and P. M. A. Sloot. A grid service for management of multiple hla federate processes. In Roman Wyrzykowski and Jack Dongarra and Norbert Meyer, Jerzy Wasniewski, editor, *Parallel Processing and Applied Mathematics: 6th International Conference, PPAM 2005, Poznan, Poland, September 11-14, 2005, Revised Selected Papers*, volume 3911 of *Lecture Notes in Computer Science*, pages 699–706, Heidelberg, 2006. Springer-Verlag.

[13] J. Kohl and P. Papadopoulos. Cumulvs User's Guide Computational Steering And Interactive Visualization In Distributed Applications. `http://www.virtc.com/Products/`.

[14] D. Kurzyniec, T. Wrzosek, D. Drzewiecki, and V. S. Sunderam. Towards Self-Organizing Distributed Computing Frameworks: The H2O Approach. *Parallel Processing Letters*, 13(2):273–290, 2003.

[15] H. Liu, L. Jiang, M. Parashar, and D. Silver. Rule-based Visualization in the Discover Computational Steering Collaboratory. *Future Generation Computer Systems*, 21(1):53 – 59, January 2005.

[16] MPICH-G home page. `http://www.niu3.edu/mpi`.

[17] B. Nichols, D. Buttlar, and J. Farrell. *Pthreads Programming A POSIX Standard for Better Multiprocessing*. O'Reilly, 1996.

[18] Open HLA Project home page. `http://sourceforge.net/projects/ohla`.

[19] OpenMP project home page. `http://www.openmp.org/`.

[20] ProActive project homepage. `http://www-sop.inria.fr/oasis/ProActive/`.

[21] Parallel Virtual Machine home page. `http://www.csm.ornl.gov/pvm/pvm_home.html`.

[22] RTI Verification Status Board. `https://www.dmso.mil/public/transition/hla/rti/statusboard`.

[23] K. Rycerz, M. Bubak, M. Malawski, and P. M. A. Sloot. A Framework for HLA-Based Interactive Simulations on the Grid. *SIMULATION*, 81(1):67–76, 2005.

[24] A. Schreiber, T. Metsch, and H.-P. Kersken. A Problem Solving Environment for Multidisciplinary Coupled Simulations in Computational Grids. *Future Generation Computer Systems*, 21(6):942–952, June 2005.

[25] Web Services. `http://www.w3.org/2002/ws/`.

[26] K. Zając, M. Bubak, M. Malawski, and P. M. A. Sloot. Towards a Grid Management System for HLA-Based Interactive Simulations. In S. J. Turner and S. J. E. Taylor, editor, *Proceedings Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003)*, pages 4–11, Delft, The Netherlands, October 2003. IEEE Computer Society.

[27] Z. Zhao, G. D. van Albada, A. Tirado-Ramos, K. Zając, and P. M. A. Sloot. ISS-Studio: a Prototype for a User-friendly Tool for Designing Interactive Experiments in Problem Solving Environments. In P. M. A. Sloot, D. Abrahamson, A. V. Bogdanov, J. J. Dongarra, A. Y. Zomaya, and Y. E. Gorbachev, editors, *Computational Science - ICCS 2003, Melbourne, Australia and St. Petersburg, Russia, Proceedings Part I*, volume 2657 of *Lecture Notes in Computer Science*, pages 679–688. Springer Verlag, June 2003.