Status of a Bridge between CORBA and Globus

Snigdha Verma [2], Jarek Gawor[1], Gregor von Laszewski[1], and Manish Parashar[2]

[1]Mathematics and Computer Science Division
Argonne National Laboratory, 9700 S. Cass Ave, Argonne, IL, 60440, U.S.A.
[2] Department of Electrical and Computer Engineering,
Rutgers University, 94 Brett Road, Piscataway, NJ 08854-8058, U.S.A
gregor@mcs.anl.gov, parashar@caip.rutgers.edu
http://www.globus.org/cog

Abstract. This paper reports on a work-in-progress aimed at designing and deploying a CORBA Commodity Grid (CoG) Kit. The overall goal of this project is to explore how to enable the development of advanced Grid applications while adhering to state-of-the-art software engineering practices and reusing existing Grid infrastructure. As part of this activity, we are currently investigating how CORBA can be used to support this software engineering task. In this paper, we outline the design of a CORBA Commodity Grid Kit that will provide a software development framework for building a CORBA "Grid domain." We also present our current experiences in developing a prototype CORBA CoG Kit that provides access to the Globus toolkit functionality to support Grid development.

1 Introduction

Developing applications as part of a problem-solving environment in the emerging national and international Grids is still a difficult task. Although Grid services exist that enable application developers to authenticate, access, discover, manage, and schedule remote Grid resources, often they may not be compatible with commodity technologies. Thus, these services may be difficult to integrate as part of the software engineering process that application developers currently adhere. Recently, a number of projects have started to investigate Commodity Grid Kits (CoG Kits) in order to overcome this problem. Developers of CoG Kits are faced with the common goal of developing mappings and interfaces between the Grid and a particular commodity technology. We believe that CoG Kits will encourage the use of Grid technologies while at the same time leveraging the benefits of the commodity technology selected. Currently, commodity technologies such as the Java platform [1] [2], Java Server Pages [3], Python [4], and Perl [5] are being considered for CoG Kits.

In the last decade distributed object technologies have been defined to ease application development in distributed environments. One of these technologies is the Common Object Request Broker Architecture (CORBA) [10] that provides an open, vendor-independent architecture and infrastructure for distributed application integration. One of the benefits of CORBA is the use of the standard protocol IIOP allowing CORBA-based programs to be portable across vendors, operating systems, and programming languages. As such CORBA delivers the necessary requirements to be seriously considered by application developers as part of the Grid infrastructure. Grid computing in contrast has so far concentrated on the creation of advanced services that allow access to high-end remote resources such as batch systems at supercomputing centers, large scale, storage systems, and large-scale instruments. Thus, it is most natural to explore how to benefit from both of these frameworks in order to provide a strong application development environment for high-end users and to create, using the appropriate CORBA terminology, a CORBA "Grid domain."

Our paper is structured as follows. First, we give a brief overview of the Grid and its architecture in order to present a solid foundation of services and protocols that we intend to integrate within our CORBA CoG Kit. We then briefly outline requirements, advantages, and disadvantageous of CORBA technologies from the point of view of Grid developers. Next, we propose an architecture that supports access to Grid functionality as part of our CORBA CoG Kit, and we present the design and implementation of a prototype. Finally, we conclude our paper and identify areas for further activities.

2 The Grid

The term "Grid" emerged in the past decade and denotes an integrated distributed computing infrastructure for advanced science and engineering applications. The Grid concept is based on the coordinated resource sharing and problem solving in dynamic multi-institutional virtual organizations [6]. Besides access to a diverse set of remote resources among different organizations, Grid computing is required to facilitate highly flexible sharing relationships ranging from client-server to peer-to-peer computing. An example for a typical Grid client-server relationship is the classical model of a supercomputer center in which a client submits jobs to the supercomputer batch queue. An example for peer-to-peer computing is the collaborative online steering of high-end applications as demonstrated in the use of advanced instruments [7][9].

Grids must support different levels of control ranging from fine-grained access control to delegation, single user to multi-user, and different quality of service levels such as scheduling, co-allocation and accounting. These requirements are not sufficiently addressed by levels the current commodity technologies, including CORBA. Though sharing of information and communication between resources is allowed, it is

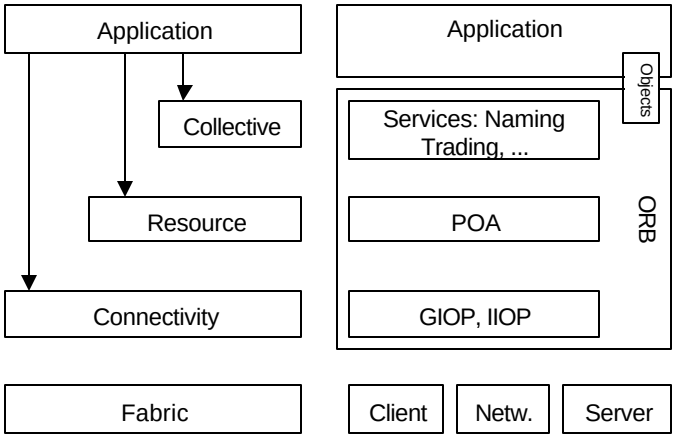

Fig. 1: The figure on the left depicts the Grid architecture. The figure on the right depicts how CORBA fits within this architecture.

Table 1: Mapping various CORBA related technologies into Grid layers.

Collective	CORBA Services: Transaction, Trading Object, Time, Security, Relationship, Query, Property, Persistent Object, Notification, Life Cycle, Licensing, Naming, Externalization, Event, Concurrency, Collection
Resource	POA
Connectivity	GIOP, IIOP, GSI, SSL
Fabric	Client, Server, Networks

not easy to coordinate use of resources at multiple sites for computation. The Grid community has developed protocols, services, and tools that address issues arising from sharing resources in peer communities. The Grid addresses security solutions that support management of credentials and policies when computations span multiple institutions, secure remote access to compute and data resources, and information query protocols that provide services for obtaining the configuration and status information of the resources. The diversity of the Grid makes it difficult to develop an all-encompassing Grid architecture. Recently, a layered Grid architecture has been proposed [6] that distinguishes a

- ***fabric layer*** that interfaces to local control including physical and logical resources such as files, or even a distributed file system;
- ***connectivity layer*** that defines core communication and authentication protocols supporting Grid-specific network transactions;
- ***resource layer*** that allows the sharing of a single resource;
- ***collective layer*** that allows to view resources as collection, and an
- ***application layer*** that uses the appropriate components of each layer to support the application.

Each of these layers may contain protocols, APIs, and SDKs to support the development of Grid applications. This general layered architecture of the Grid is shown in the left part of Fig. 1.

3 CORBA for Grid Computing

Our objective is to design and implement a CORBA CoG Kit that provides users and developers in a virtual organization with access to Grid Services using the CORBA distributed computing technology. Among the advantageous of CORBA that makes it suitable for development of the CoG Kit are the following:

- *Standard Protocol*. CORBA uses a standard protocol (IIOP) that provides the basis for its portability. The use of this protocol provide the basis for
  - *Vendor independence*. CORBA is adopted by many vendors and in principle should be portable between vendors.
  - *Language independence*. Interfaces between clients and servers are defined in an interface definition language providing multi-language and multi-platform support.
  - *Operating system transparency*. CORBA is supported on many operating systems.
- *Legacy Integration*. Legacy applications can be easily cast as CORBA objects.
- *Location Transparency*. CORBA objects and services can be invoked in the same way by software residing anywhere on the network. The fact that the software invoking the component on a potential different computer is hidden.

- *Network Transparency*. The CORBA platform hides the network from the application programmers.
- *Secure Communication*: The integration of SSL over the IIOP protocol allows secure communication between client and remote objects.

The separation of "interface from implementation" as part of the CORBA platform provides the essence of transparency and independence for application programmers. Thus, even though an interface is precisely defined, its implementation may vary. One of our tasks within this project is to evaluate if this transparency can be proven in practice while evaluating several existing ORBs for the suitable integration with Grid services and components. Multiple strategies exist for using CORBA in a Grid environment and provide additional value for application developers.

Although, as pointed out in [7], CORBA could be viewed as part of the application layer that is based on top of the existing Grid infrastructure, we need to enhance this view as CORBA provides additional features capable of supporting each of the layers of the Grid architecture. We have illustrated this in Fig. 1 on the right side. As part of the protocol layer we can access IIOP and GIOP; as part of the resource layer one can argue that POA provides the basis for defining objects/resources in a portable way; as part of the collective layer many CORBA services provide a transparent access to objects that get accessed by the applications. To support this different view we list some of the scenarios demonstrating how Grid users can use CORBA:

1. A Grid application developer may want to set up ORBs on a set of remote resources that are part of the Grid in order to enable a distributed computational environment based on CORBA objects, services and components.
2. A Grid application developer may want to have additional security restrictions defined by Grid security protocols and authentication frameworks, on objects accessible within an ORB.
3. A CORBA application developer may want to access Grid services such as the submission of jobs to supercomputers or the access to files as part of Data Grids.
4. A Grid application developer may want to access CORBA services and functionality that enhance the application.

While (1) and (3) can easily be mapped into the application layer, it is obvious that (3) and (4) are best represented as shown in Fig. 1. Since (1) can easily be implemented with the current Globus Toolkit, we will not discuss it further in this paper. Although it will be a useful feature to enable object access with the help of, for example, Grid security as provided by the Globus toolkit, we analyzed that an implementation would require significant modifications of the internals of the ORB and potentially prevent the interoperability between different vendors. Nevertheless, we believe such an activity must be seriously considered if a reusable protocol layer within an ORB can be provided in a in a plugin fashion on top of SSL.

Our primary focus in this paper is on providing access to existing Grid services through CORBA. This is achieved by providing Grid domain interfaces to encourage portable access to Grid services as described in the next section.

4 CORBA Grid Domain Interfaces to Grid Services

In the process of delivering a CORBA CoG Kit we have decided to concentrate our effort on the creation of mappings and interfaces to Grid services provided by the Globus Toolkit. The most important services for our initial activities include security, remote job submission, and information management. These services are elementary and essential for enabling computational Grids, since they provide the foundation for building more advanced Grid services. Globus provides an authentication service as part of the Grid security infrastructure (GSI) [15], an information service called Metacomputing directory service (MDS) [12], and a job submission service called Grid Resource Allocation manager [13]. We summarize each service briefly before we describe how to access them from CORBA.

4.1 Grid Information Service

The Metacomputing Directory Service (MDS) provides the ability to manage and access information about the state of a Grid. As such it enables read access to entities such as computer, networks, and people. The current implementation of MDS as part of the Globus Toolkit is based on a distributed directory based on LDAP technology [16]. A high-end application can access information about the structure and state of the system through the uniform LDAP API. The information is organized in MDS as well defined collection called entries. The entries are organized in a hierarchical tree structured name space called Directory Information Tree (DIT). Any of the MDS entry can be referred by its unique name called Distinguished Name, which is constructed by specifying the path from the root to the entry being named. These entities represent an instance of an object. The information in an entry is represented by one or more attributes consisting of name and corresponding values. The attributes depend on the object type that the entity is representing. The object type information is encoded in the MDS Data Model. This data model specifies the data hierarchy and the object classes used to define each type of entry. The access and usage of a Grid information service within a Grid must consider several cases.

4.1.1 Grid Integration of a Common Naming Service

Many CORBA implementations provide naming services that are based on LDAP directories. An example for such use can be found within JNDI, which can be used to access LDAP directories from Java. Within JNDI a COS naming service is provided [17]. The advantage of such a service is that all objects can be stored in the same location as other Grid-related objects, potentially reducing the overhead costs for maintaining the Grid information infrastructure.

```
module MDS {
        struct Result {  string id;
                         sequence<string> value;
        };
        typedef sequence<Result> MDSResult;
        typedef sequence<string> Attributes;
        struct ListResult {  string id;
                             MDSResult value;
        };

        typedef sequence<ListResult> MDSList;

        interface MDSServer {
                exception MDSException {
                        string mdsMessage;
                        string ldapMessage;
                };

                void connect (in string name, in long portno,
                                in string username,  in string password)
                        raises (MDSException);

                void disconnect() raises (MDSException);

                MDSResult getAttributes(in string dn) raises (MDSException);

                MDSResult getSelectedAttributes (in string dn, in Attributes attrs)
                                raises (MDSException);

                MDSList getList(in string basedn) raises (MDSException);

                MDSList search( in string baseDN, in string filter,
                                in long searchScope)
                        raises (MDSException);

                MDSList selectedSearch (in string baseDN, in string filter,
                                in Attributes attrToReturn,
                                in long searchScope) raises (MDSException);

        };
}
```

Fig. 2: The interface for the MDS

4.1.2 Simple Grid Domain Interfaces for Accessing the MDS

Grid applications frequently query the MDS for current information about the Grid in order to use this information for steering of Grid applications. Thus it is advantageous to provide a simple interface supporting the following:

1. Establishing connection to the MDS Server
2. Querying the MDS Server
3. Retrieving results obtained from the MDS Server through a query
4. Disconnecting from the MDS Server

The implementation of the CORBA MDS Server object provides this functionality. It accesses the Globus MDS using JNDI libraries and essentially replicates the functionality provided by the Java CoG Kit [1]. Fig. 2 shows the simple IDL we have designed for this purpose. The MDS search result and the lists of attributes are stored as JNDI *NamingEnumeration* data. As CORBA is a language-independent middleware, it is necessary to map the *NamingEnumeration* into a generic data type. In the IDL file for the CORBA MDS Server object, a number of structures are defined (e.g. *MDSResult, MDSList, Result)*.

```
GlobusMachineTrader {
        struct MachineType {
                string dn;
                string hn;
                string GlobusContact;
                long   freenodes;
                long   totalnodes;
        };

        typedef sequence<MachineType>  MachineTypeSeq;

        …

        interface GetMachineInfofromMDS {
                void update_seq();
                void initialize_trader();
                void update_trader();
                void refresh_trader();
        };
};
```

Fig. 3: A simple example for a CORBA trader to access selected MDS information

4.1.3 Grid Domain Trading Service

A CORBA trader is used to store advertisements for services. In the CORBA CoG Kit, we can use a CORBA trader to store service offers for Grid related services and resources of a particular type. The service type will determine which properties are used to describe a service offers. To illustrate this with an example, we assume that we like to create a trader that returns information about the number of free nodes of a compute resource. Such information can be easily exported to a trader from the MDS through either direct access or through the use of the interface in Section 4.1.2. Fig. 3 illustrated such an interface in more detail. This interface could be used as an example for providing more sophisticated custom-designed trading services. It is obvious that such a trader could provide bridges between different information sources and build the basis for more sophisticated information service within the Grid. Such a trading server has been successfully prototyped and implemented as part of [19].

4.2 Accessing Grid Security Mechanisms

Being able to access Grid security mechanisms is an essential part of the CORBA CoG Kit. We base our current implementation on the Grid Security Infrastructure (GSI) from the Globus Toolkit. GSI [15] is based on protocols for authentication and communication, building on the Transport Layer Security (TLS) protocols. GSI addresses single sign on in virtual organizations, delegation, integration with local security solutions, and user-based trust relations [6]. As such, GSI is used to overcome cross-organizational security issues.

As indicated in Section 3, one can integrate Grid security features at various levels of the CORBA architecture. In order to enable portability between ORBs, in this effort we have not considered the modification of the protocol stack, but have placed all provisions for enabling access of the Grid security to allow authentication to Grid services such as MDS and Job submission, on the client side. This strategy has been proven successful as part of the Java CoG Kit [1].

4.3 Job Submission in a Grid

Only after the appropriate authentication through the use of GSI in an instantiation of a Grid has been performed is it possible to submit jobs via CORBA to Globus services such as GRAM. In our prototype implementation we are providing two ways of accessing Grid job submission.

4.3.1 SimpleGlobus.run

The Globus Toolkit provides a shell command called “globusrun” enabling simple access to remote job submission. We are providing simple interface that allows clients to access the functionality globusrun provides. simpleGlobus.run provides the ability to specify an execution context with the help of a resource specification language [13]. In general it allows the specification of a program with a set of parameters to be executed while an option exists the IO can be redirected to the client starting the program. This redirection is controlled by the special set of GlobusArguments that have to be set prior to accessing GlobusRun. The actual submission of the job is performed currently through full delegation, requiring at least one server of the ORB to have a version of Globus installed that provides the original shell script. With this simple implementation users have the ability to access in a straightforward fashion a subset of the capabilities for job submission within the Grid. In Fig. 4 we show the simple IDL for accessing the Grid.

```
interface simpleGlobus {
        void setRsl(in string rsl);
        void setGlobusArguments(in string rsl);
        void setStdout(in string stdout);
        void setStderr(out string stderr);
        void setStdin(in string stdin);
        boolean run();
};
```

Fig. 4: The IDL for accessing "globusrun" on clients within the ORB.

4.3.2 GRAM

Although the simpleGlobus interface provides a straightforward implementation of basic capabilities to perform remote job submission within a Grid, it fails to give the application developer the control provided by the existing GRAM capabilities. The GRAM service is used for allocating jobs on different computational resources. It consists of– a GRAM client library, gatekeeper, an RSL parsing library, a Job Manager, a GRAM reporter. Together GRAM and GSI for authentication and authorization application can not only submit jobs, but also observe their status (globusrun is a convenience function to start jobs through GRAM from the commandline). The process of remote job submission includes the following steps. Initially the application authenticates with the Gatekeeper at the remote site by executing calls to GSI libraries. Next one specifies the resources, the binary for execution, and a callback function that react upon state changes to the job. The Globus gatekeeper at the remote end mutually authenticates the user and resource, determines a local user name for the remote user and starts a job manager. The job manager creates the actual process requested by the

user. It submits the resource allocation request obtained from the RSL parsing library to the local resource management system. If there is no local resource management system then a simple fork is performed. Once the process is created, the job manager monitors the state of the process and notifies the callback function whenever there is a state transition or process termination. The responsibility of job manager ends once the process has terminated. **Fig. 5** demonstrates the various ways in which a CORBA client may provide job submission within a Grid.

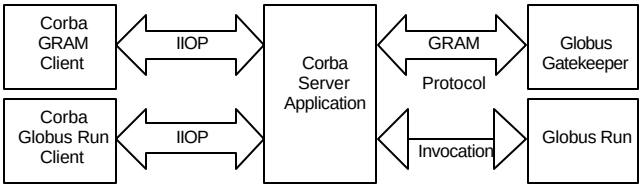

Fig. 5: Accessing a Globus Gatekeeper from a CORBA client.

5 Applications

Many applications can benefit from a CORBA CoG Kit. One example is the Numerical Propulsion System Simulation (NPSS) [19] as part of NASA IPG that provides an engine simulation using computational fluid dynamics (CFD) in 0-to 3-dimensional engine component models responsible for examining aerodynamics, structures, and heat transfer. In previous studies it has been shown that NPSS's engine components can be encapsulated by using CORBA in order to provide object access and communication from heterogeneous platforms while at the same time coordinate modeling runs across Globus. As part of this task a large number of NPSS jobs (1000+) are submitted from a desktop interface returning the output to that same interface via the CORBA CoG Kit. The longer-term goal includes deploying more computationally intense (3-dimensional) NPSS jobs across the Globus-enabled NASA Information Power Grid (IPG). The benefit for this project is based on being able to access Globus functionality directly from a CORBA application.

Other examples include the possible control of advanced scientific instruments such as radio telescopes and synchrotron rings, since part of their commercially available control infrastructure is provided as access through CORBA objects. At many of these installations it will not be possible to install Globus server side software but only to interface to it as a client.

6 Status

Currently our integration with Grid Information Services is completed. We have concentrated on a direct interface to the MDS. Previous effort has demonstrated that it is straightforward to generate specialized trading services in CORBA. Once we have identified a suitable set of requirements for information for the targeted applications, we will provide for this application domain a specialized trading service.

We have made significant progress in the delivery of services supporting the remote job submission. We ran into technical problems with several ORBs and are currently discussing these issue with the vendors/developers of these ORBs. All problems are related to the integration of security features. We are confident that these problems will be solved together with the vendors, providing us the possibility for job submission while using GSI. We hope to also extend the security within the Grid Information Services while making use of SASL once the new version of Globus is available. Additional future, activities will include identifying a suitable set of test cases for verifying the usefulness of the CORBA CoG Kit. We will test case the implementation as part of the DISCOVER project [9].

7 Conclusion

We are in the process of defining and implementing a CORBA CoG Kit that provides a framework to enable existing Grid Computing Environments and CORBA Service Provides to interoperate. Since CORBA is designed with a distributed environment in mind, has support from many vendors, provides transparency on the levels of language, operating system, network, and protocol, it is an ideal candidate for applications programmers to develop Grid based applications. Providing a CORBA Grid domain will allow the easy integration of additional Grid services and functionality within these applications. The demand for such a CoG Kit has been expressed by various projects ranging from the creation of CORBA based control systems for advanced instruments to the computational steering of fluid dynamics codes. Our future effort will concentrate on enabling applications to combine services provided by Globus, with the collaborative monitoring, interaction, and steering capabilities provided by DISCOVER [9]. For example a scientific simulation application can use CORBA CoG Kit to discover the available resources on the network, use the GRAM Service provided by CoG to run his simulation on the desired high end resource, and use DISCOVER web-portals to collaboratively monitor, interact with, and steering the application.

Acknowledgments

We thank Ian Foster for his valuable discussions. We thank Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, David Woodford, Isaac Lopez, Gregory J. Follen, Richard Gutierrez in their efforts to provide a C++ based CORBA interface for the NPSS application performed at NASA Glenn Research Center. We thank Robert Griffin for the development of a C++ based implementation of the Grid information services interface. Additionally, we thank Nell Rehn and Peter Lane for efforts in helping to develop the Java CoG Kit.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation under Grant Number ACI 9984357 (CAREERS) awarded to Manish Parashar, and by the NASA Information Power Grid program.

References

[1] G. v. Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, 2001 To appear. http://www.globus.org/cog/documentation/papers/cog-cpe-final.pdf.

[2] G. v. Laszewski, I. Foster, J. Gawor, W. Smith, and S. Tuecke, "CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids," in *ACM 2000 Java Grande Conference*. San Francisco, CA, 2000, pp. 97--106, http://www.mcs.anl.gov/~laszewsk/papers/cog-final.pdf.

[3] "The Grid Portal Development Kit," 2001, http://dast.nlanr.net/Features/GridPortal/.

[4] "The Python CoG Kit," 2001, http://www.globus.org/cog.

[5] "The Perl CoG Kit," 2001, http://hotpage.npaci.edu.

[6] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabeling Scalable Virtual Organizations," *International Journal of Supercomputing Applications*, vol. To appear., 2001 http://www.globus.org/research/papers/anatomy.pdf.

[7] Y. Wang, F. D. Carlo, D. Mancini, I. McNulty, B. Tieman, J. Bresnahan, I. Foster, J. Insley, P. Lane, G. v. Laszewski, C. Kesselman, M.-H. Su, and M. Thiebaux, "A high-throughput x-ray microtomography system at the Advanced Photon Source," *Review of Scientific Instruments*, vol. 72, pp. 2062-2068, 2001.

[8] I. Lopez, G. J. Follen, R. Gutierrez, I. Foster, B. Ginsburg, O. Larsson, and S. Tuecke, "Using CORBA and Globus to Coordinate Multidisciplinary Aeroscience Applications," presented at Proceedings of the NASA HPCC/CAS Workshop, 2000.

[9] "The DISCOVER Project," 2001, http://www.caip.rutgers.edu/TASSL/Projects/DISCOVER/Documents.html.

[10] “CORBA: Common Object Request Broker Architecture”, http://www.omg.org.

[11] CORBA 2.0/IIOP Specification. http://www.omg.org/corba/c2indx.htm.

[12] The Globus MDS. http://www.globus.org/mds.
[13] The Globus GRAM. http://www.globus.org/gram.
[14] J. Bester, I. Foster, C. Kesselmaz, J. Tedesco, and S. Tuecke "GASS: A Data Movement and Access Service for Wide Area Computiing Systems"
[15] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. *A Security Architecture for Computational Grids*. Proc. 5th ACM Conference on Computer and Communications Security Conference, pg. 83-92, 1998.
[16] Netscape Directory and LDAP Developer Central http://developer.netscape.com/tech/directory/index.html.
[17] JAVA Naming and Directory Interface (JNDI) http://java.sun.com/products/jndi.
[18] IONA Technologies, ORBIX 2000 http://www.orbix.com/products/orbhome.htm
[19] Numerical Propulsion System Simulation (NPSS) http://hpcc.lerc.nasa.gov/npssintro.shtml .