Open Collaborative Grid Service Architecture (OCGSA)

Kaizar Amin
Argonne National Laboratory
Argonne, IL, USA

Sandeep Nijsure
Argonne National Laboratory
Argonne, IL, USA

Gregor von Laszewski
Argonne National Laboratory
Argonne, IL, USA

Abstract

In this paper we introduce a new architecture, called *Open Collaborative Grid Services Architecture* (OCGSA); that serves as the building block for designing collaborative services within Grids. The framework is based on the Open Grid Services Architecture (OGSA) that combines aspects of Grids with Web services. Using OGSA enables users to access advanced Grid features such as lifetime management, notification, and security that are otherwise not available in Web services. Our framework provides high-level Grid services that can be integrated within future collaborative applications, hence making them automatically Grid aware. Besides the definition of such useful advanced collaborative services, we use this architecture to validate the approach taken in OGSA and verify whether necessary collaborative services can be implemented with the help of OGSA. Use of these services will also enable the creation of peer-to-peer collaborative infrastructures.

1. Introduction

The term *Grid computing* [1] is commonly used to refer to a distributed infrastructure that promotes large-scale resource sharing in a dynamic multi-institutional virtual organization. Just as the electric power grid provides pervasive access to electric power, the computational Grid provides ubiquitous access to a large collection of compute-related resources and services. Grid computing raises a wide range of issues as a result of large-scale resource sharing among separately administered institutions. These issues include security, resource management, and resource discovery. Recently, the Grid architecture has made a noticeable shift toward a *services-oriented framework* [2]. A *Web service* is a platform-independent software component that is described by using a service description language [3] and provides functionalities to publish its interfaces to some directory or registry of services. These published services can be discovered by querying the service directories or registries and can be invoked by using appropriate binding protocols [4], thereby providing networked access to some functionality otherwise not available locally. The services-oriented concept lends itself to an architecture that requires interoperability across heterogeneous platforms and exposes all or part of its applications on different machines, platforms, and domains [5]. Hence, the Open Grid Services Architecture (OGSA) [6] was developed to provide an extensible framework of services that support Grid functionalities by combining the realms of Web services and Grid technologies.

Web services address the discovery and invocation of *persistent* services, whereas Grid technologies need to tackle issues related to creation, discovery, and destruction of *transient* service instances. Therefore, the OGSA introduced the notion of *Grid service*: a (potentially transient) Web service that provides a set of well-defined interfaces and that follows some standard conventions [6]. OGSA provides a framework that supports

Instance Creation : Grid service *Factories* can be requested to create new, reliable, and transient service instances. It provides a mechanism to individually identify these instances by associating some state information with each service instance.
Lifetime Management: The lifetime of these transient service instances can be negotiated with the service Factory at the time of their creation. It may also support soft-state lifetime management, thereby avoiding the need for explicit destruction of service instances.
Registration and Discovery: Allows the registration and discovery of transient service instances. It provides an efficient query mechanism for the client using the *FindServiceData* operation of the Grid service against the metadata maintained by that registry. It also supplies a Web Service Inspection document that contains details about all the services maintained by that registry.
Notification: Establishes a publish/subscribe style of notification mechanism between service providers and service requesters, thereby supporting an asynchronous peer-to-peer style of communication.
Security: Handles the required authentication and authorization policies defined by different application and domain administrators.

Sophisticated applications that can be controlled and steered by multiple users in an interactive fashion can benefit from the functionalities provided by OGSA. For example, consider a research collaboratory that has a suite of advanced scientific applications as a collection of Grid services. End users can invoke these services individually or collaborate in groups, exchanging their data sets, interactively manipulating their input to these services, and participating in collaborative discussions based on output results. Although OGSA provides all the constructs (core services) to implement such platform-independent research collaboratories, considerable development effort is involved in each case. Every collaborative application requires certain standard components at a minimum. Rather than re-developing these components in every project, it is worthwhile to invest some effort in identifying a common infrastructure that provides all the basic components required for collaborative applications and can be inherited by any stand-alone application with minimum effort to transform itself into a collaborative Grid service. This paper proposes a framework that builds on top of the OGSA infrastructure, providing a set of services that can be used by any collaborative application. We call this new framework *Open Collaborative Grid Services Architecture* (OCGSA).

Section 2 provides the architectural details of the OCGSA framework, discussing the principles of individual components. Section 3 provides a brief overview of several different projects that address the issue of a scalable collaborative architecture. The paper concludes with a summary and synopsis of ongoing research to develop the OCGSA framework.

2. OCGSA Framework

The Open Collaborative Grid Services Architecture composes a set of common components that can be easily customized for individual applications. The OCGSA framework enable users to form ad-hoc collaborative groups by interacting over a set of predefined notification topics. It provides appropriate lifetime management for individual groups, offers an advanced discovery mechanism for service instances, and establishes sophisticated security mechanisms at different levels of the application. Figure 1 shows the OCGSA framework depicting the common components for collaborative applications. In the subsequent sections, we describe individual components of this framework, providing an overview of their interaction with peer components.

2.1 Collaborative Grid Service

The OGSA specification defines a Grid service as any Web service that conforms to a predefined interface. In other words, a Web service is a Grid service if it implements the *GridService portType* [6]. The GridService portType provides a base interface for the service instance to act as a Grid service: providing maintenance of its metadata, lifetime management, and support for sophisticated security. OGSA allows Grid service developers to define customized metadata for individual applications in addition to the standard set of required service data elements. In a collaborative environment every Grid service instance warrants certain functionality and metadata for the maintenance and discovery of individual service instances (groups). Rather than redefining this metadata information for every collaborative Grid service, we introduce the notion of a *collaborative Grid service*: a Grid service with a set of predefined metadata elements and behaviors. In analogy with the GridService portType, we define a *CollaborativeGridService portType* that defines all the base interfaces that are required for a collaborative Grid service instance. Among other things, a CollaborativeService portType defines metadata regarding the creator of a group, name of the group, description of the group, current members of the group, and level of event archiving desired. It also provides the interface to handle security policies at different levels of the application (application level and group level), basic community chat facilities, an adequate presence management for users, and capabilities for interactively controlling and steering the application.

A collaborative Grid service implements the GridService portType and the NotificationSource portType. This allows the collaborative service to generate publish/subscribe style notification messages to its clients by means of which they can interactively communicate. The NotificationSource portType is provided by OGSA; however, the OCGSA framework provides constructs in the *Collaborative Grid Service Descriptive Language* (cgsdl) by which a developer is able to predefine a set of notification topics that will be supported by collaborative Grid services (see Figure 2). The collaborative Grid description language is an extension to the *Grid service description language* (gsdl) defined by OGSA and is used to describe details of a collaborative Grid service. The code generator will generate the necessary stubs for this definition, and the application -specific details for each notification topic can be implemented by the developer.

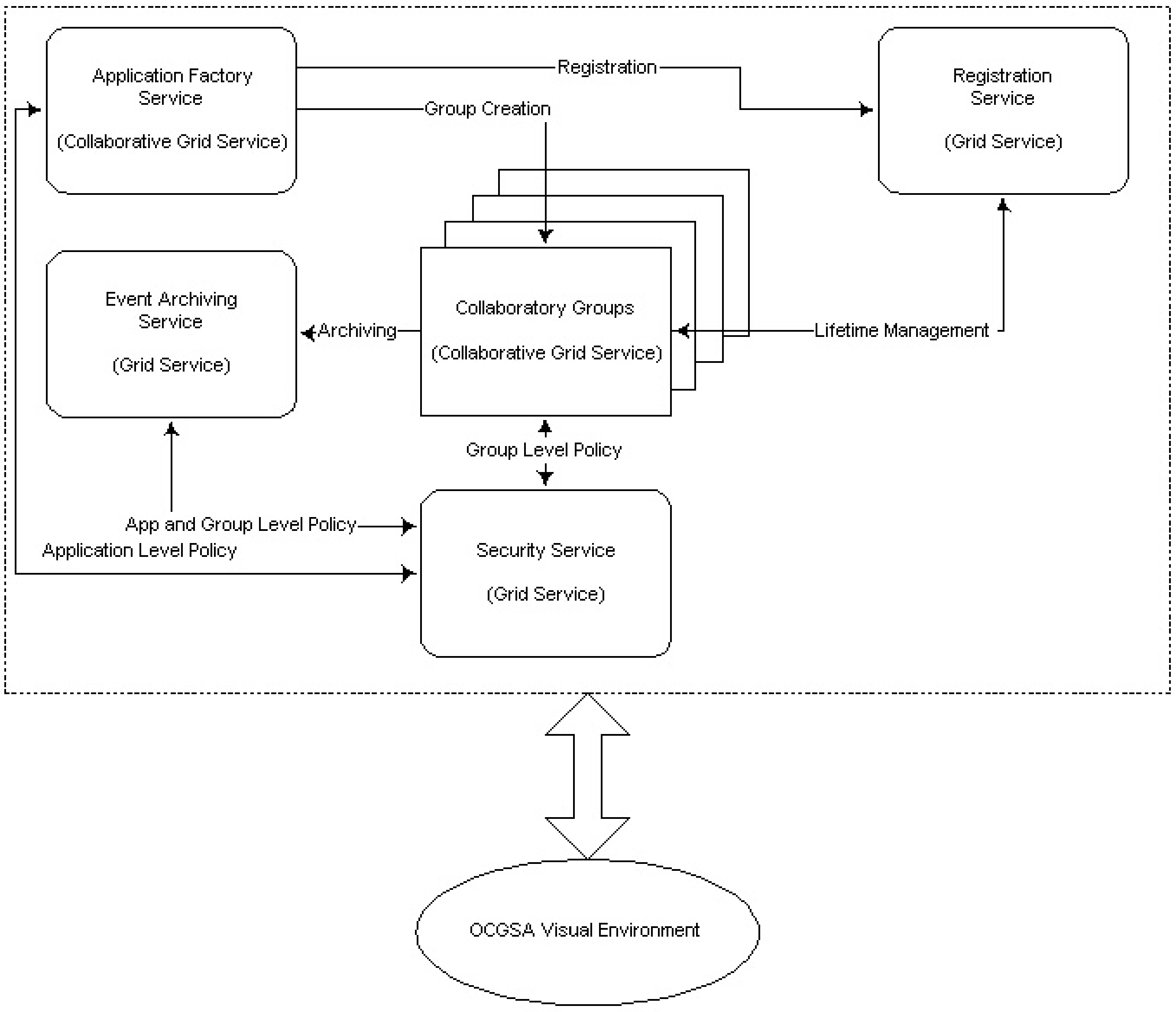
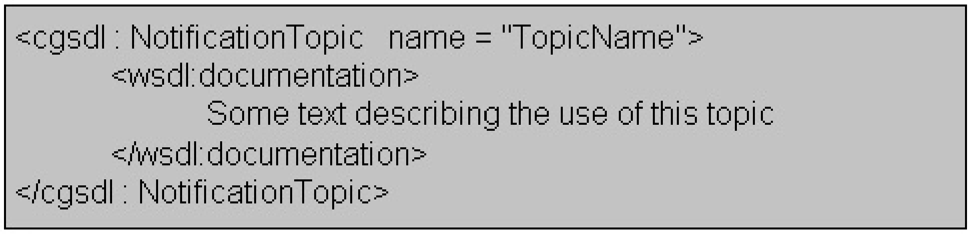


Figure 1. OCGSA Framework

Although OCGSA defines a collaborative Grid service as its primary component, it is not sufficient by itself to provide complex interactive collaboratories. OCGSA defines other services such as registration, event archiving, security, a visual environment, and the pattern for their interaction to support sophisticated interactions. The following sections describe each of these services in detail.



```
<cgsdl : NotificationTopic   name = "TopicName">
        <wsdl:documentation>
                Some text describing the use of this topic
        </wsdl:documentation>
</cgsdl : NotificationTopic>
```

Figure 2. CGSDL

2.2 Registration and Discovery

At the heart of any service -based architecture is the concept of service registration and discovery. The OGSA framework provides a registration service that implements the Registration portType interface along with a set of metadata elements that contain information about all the services registered with it. Using the *FindServiceData* operation of the *GridService* portType, clients can discover the registered services from the information contained in the registry meta -data. The Registration service also provides a Web service inspection (WS -Inspection) document that contains information about the various services registered with it. The OCGSA framework requires that every collaborative service be associated with a registration service. A single registration service can register instances of different collaborative services. The registration service used in OCGSA is a registration Grid service capable of supporting XPath/XQuery queries. Support for such a sophisticated querying mechanism is essential because OCGSA enables arbitrary users with appropriate credentials to query the registration service based on complex criteria and identify the service instance (group) that they would like to join from the set of services returned by the query. Using a query language to locate services enhances the ability to discover Grid services in the OCGSA framework: services can be discovered not only by service names but by a variety of other metadata information. Although a single registration service should suffice , the OCGSA framework is capable of supporting a set of registries, organized in a hierarchical or peer -to-peer fashion where a query submitted to any registry in the framework can be relayed to other registries if all the information is not present locally. The registration service must constantly communicate with the individual instances to maintain the lifetime management information up to date for the respective services and must automatically clean the entries for defunct groups.

2.3 Event Archiving Service

Another essential service for a collaborative framework is an event archiving service, which logs the messages or events communicated between online users of a group instance into a persistent database. Users that join a particular group at a later time can retrieve these events, thereby updating themselves about the group's prior activities. The event archiving mechanism also enables the group members to insert checkpoints into the event database and replay the entire or sections of the archived collaborative activity when the group is off -line. In the OCGSA framework, we assume an event to be a <time, message> tuple. Message archiving can be implemented by using technologies such as Java Message Service (JMS) [7] or XML databases (XML:DB) [23]. However, OCGSA offers a sophisticated archiving service that functions more than a simplistic event database. The event archiving service in the OCGSA framework will be a Grid service, thereby efficiently handling the lifetime management policies of individual archives that can be set and manipulated by creators of corresponding group instances. Further, the event archiving service will implement the same security policies as implemented by the collaborative Grid services, hence forcing appropriate authorization policies at the application level and individual group level (See section 2.4). This ensures that a malicious user lacking the proper credentials, who was denied access to an active group earlier cannot have access to the communication archives when the group is off -line. Similar to discovering a collaborative Grid service, it is extremely important to efficiently and conveniently discover event archives. On fundamental level, the archiving system must support *QueryByName* style of information retrieval indicating if an archive for a particular group exists. Nevertheless, support must also be built to enable advanced queries of meta -data values of individual group archives similar to that of the OCGSA registration component. Hence, end users will be able to discover archives hosted by a set of archiving services based on complex queries regarding group names, lifetime validity, security authorization, and similar criteria.

In order to implement the functionality mentioned above, OCGSA provides an Event Archiving Grid service that provides an interface to XML:DB [23]. Hence, the Grid service can take care of the security and lifetime management issues, and the XML:DB (NXD, native XML database) provides the functionality for the data storage and retrieval using XPath queries. The XML:DB can be embedded within the Grid service, or can be located on an individual machine. The Event Archiving service can be considered as a specialization of a generic DataStorage Grid service that provides a Grid service interface to a native XML database (NXD) implementing all the generic operations required to manipulate the XML database that can store any XML document including event logs.

2.4 Security Service

Collaborative Grid services may be accessed by different groups of people from different institutions and domains. Trust levels within such groups are not as high as in a single real organization. Most of the security constructs such as authentication, delegation of rights, confidentiality, and message integrity have been specified in the OGSA security draft. Nevertheless, the area where OCGSA security must build over the OGSA security is access control (authorization). The OCGSA framework specifies two levels of authorization. On the first level, called the *application level* , the application service provider dictates the policy about who can access the service based on the identity of the requester and the nature of the request. Subscription to notification topics and access to the event archiving service are based on this policy. A part of this policy decides who can create and register a new group. The second level is the *group level* access control. The group creator specifies who can join the group and the privileges of individual users within that group. For example, the group creator might authorize some users to only listen to the communication and not actively participate. The challenging aspect of OCGSA security is the ability to define and enforce such policies dynamically at runtime without any prior configuration support from a specific system administrator.

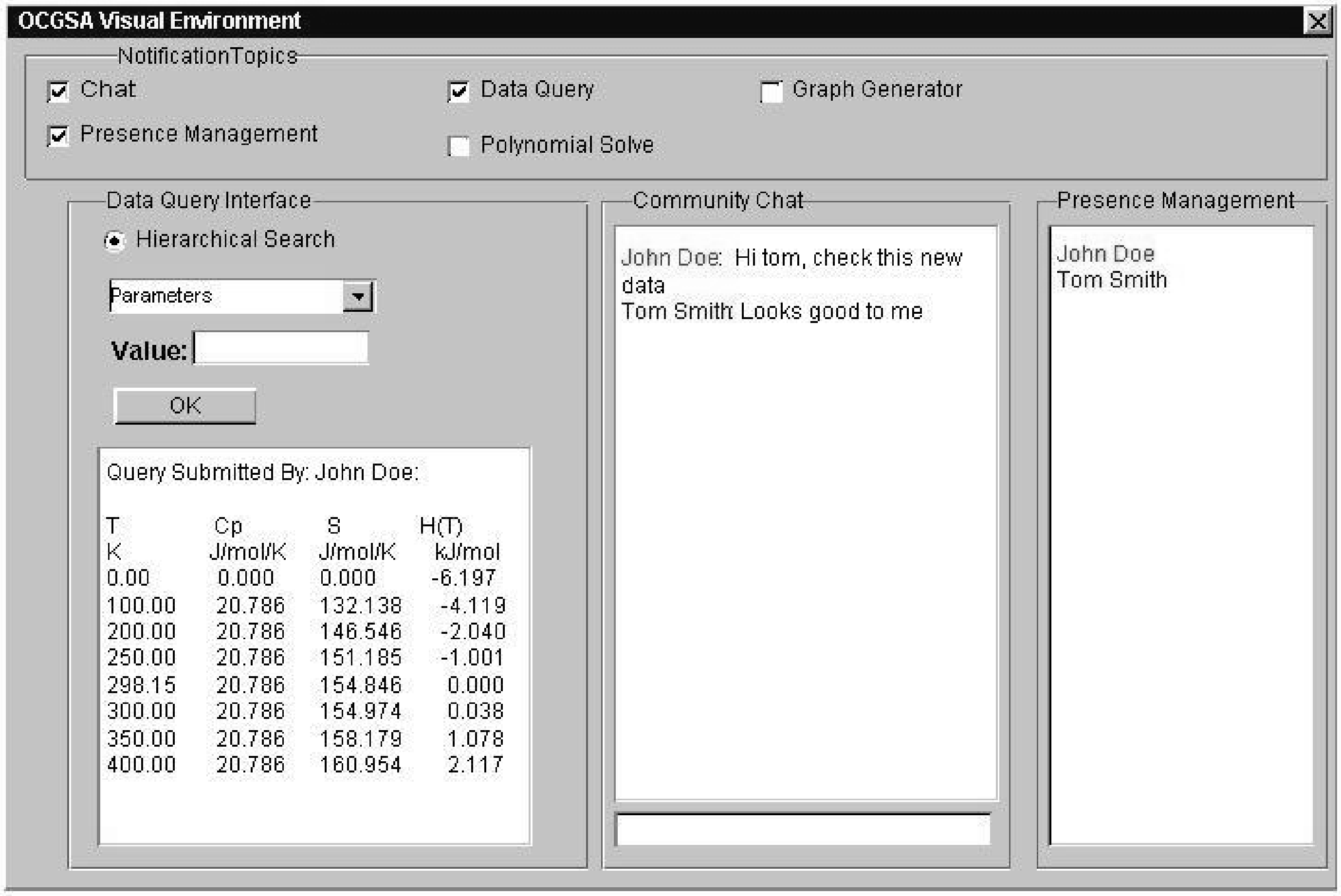
The *Community Authorization Service* (CAS) of the Globus Toolkit [15] provides similar features to those desired by the OCGSA security service. Nevertheless, CAS requires that the security policies be defined statically and is maintained by a CAS administrator. The OCGSA security service provides an extension to the functionality offered by the CAS by specifying the policies dynamically. Hence, any user that desires to join a collaborative group communicates with the OCGSA security service by submitting his credentials in the form of X509 digital certificate. The security service then maps the user certificate to the application and group-level policies and accordingly rejects or grants the user a delegated proxy X509 certificate. The user uses this proxy certificate to communicate to the collaborative service instance where his proxy credentials are mapped against group access rights.

2.5 Visual Environment

While security, registration, discovery, and archiving are important aspects for every collaborative system, the success of the collaboration is finally determined by the ease of use and effectiveness of its visual interface. Since OCGSA proposes an open extensible framework, whereby individual collaborators can form communities and interact at different levels, it must also provide with an interface environment that can be conveniently modified based on user preferences. OCGSA visual interface follows a modular approach, whereby every notification topic declared by the collaborative Grid service will be represented as an individual *topic module*. Hence, at the user interface level, subscribing or unsubscribing to a notification topic involves adding or removing the corresponding module, respectively. The application developers can expand on this framework providing rich visual interfaces to any collaborative Grid service. A wide range of implementation technologies are available to this end. For example, applications using the Java Swing technology can implement each of their topic modules as *panels* that can be loaded dynamically using custom *classLoaders*. Another promising technology that can support our flexible framework is the Apache Jetspeed Portal technology [9], whereby every topic module can be implemented as an individual portlet. This technology has an additional advantage over several other peers because Jetspeed portlets are rendered in Web browsers, thereby eliminating the need to install complex and bulky graphical rendering engines and packages. Irrespective of the implementation technology employed by the user, the OCGSA visual environment provides the pattern for communicating with the security service to obtain the appropriate credentials for every new group joined by that user. All further communications in that group are done by using the acquired credentials. Figure 3 shows a sample prototype of the OCGSA visual environment., including the topic modules for the *data query interface*, *chat*, and the *presence management*. If the user unsubscribes from any of these topics, a different interface without that topic module would result.

3. Related Work

A large number of applications have been built for interactive steering and community collaboration. In this section, we discuss applications where scope and goals are similar to those of our framework. The *Gateway* [13] system presents a three-tier architecture, whereby multiple clients connect to a high-end computing resources managed by the Globus Toolkit [15], via CORBA-based middleware. The Gateway system provides the end user with a network of distributed severs to create a visual programming environment. A similar approach is also employed by the *SciRun* [16] system. SciRun uses visual programming to allow interactive steering and construction of large scale computational systems. A Web based multitier environment for collaborative steering of scientific applications is presented in [17]; this approach uses a network of Java interaction servers as the middleware. The *Access Grid* [18] uses the Grid technology to create a collaborative environment with multimedia displays, interactive presentation systems, and interfaces to other visual environments. The *Collaboratory Interoperability Framework* (CIF) [19] provides a unifying infrastructure for diverse collaborative applications. It has a philosophy similar to that of OCGSA, building on a set of common components and promoting the reuse of common interfaces. The *Habanero* [20] framework provides a set of Java APIs to create collaborative applications in Java. *Groove* [21] is a peer-to-peer collaborative system that enables users to communicate and interact with one another and in groups in an ad-hoc fashion.



[✓] Chat
[✓] Presence Management
[✓] Data Query
[ ] Polynomial Solve
[ ] Graph Generator

(●) Hierarchical Search
Parameters
Value:

```
Query Submitted By: John Doe:

T          Cp        S         H(T)
K         J/mol/K   J/mol/K    kJ/mol
0.00       0.000    0.000     -6.197
100.00    20.786   132.138    -4.119
200.00    20.786   146.546    -2.040
250.00    20.786   151.185    -1.001
298.15    20.786   154.846     0.000
300.00    20.786   154.974     0.038
350.00    20.786   158.179     1.078
400.00    20.786   160.954     2.117
```

John Doe: Hi tom, check this new data
Tom Smith: Looks good to me

John Doe
Tom Smith

Figure 3. OCGSA Visual Environment

4. Summary and Ongoing Research

This paper discusses the Open Collaborative Grid Services Architecture (OCGSA), providing an overview of its core components. OGSA provides a strong Grid services -based framework for applications with loosely coupled components distributed over multiple platforms and administrative domains. Using the OGSA infrastructure, users can create, register, discover, and invoke potentially transient Grid services, thereby exploiting its highly distributed design. The open architecture of OGSA permits enough flexibility to extend its core components into a complex application. Every class of application has a base of common components over which it can be extended. Rather then building these base components for every class, it is worthwhile to develop a core set of components and reuse them in other applications. This paper concentrates on such a core set of components required for a generic collaborative application. The OCGSA framework extends the notion of Grid services into the collaborative domain, introducing the concept of a collaborative Grid Service. The collaborative Grid service supports different levels of security policies, basic community chat, user presence management, and metadata information regarding the collaboratory. The framework also supports the implementation of transient event archives in a secure infrastructure. The archiving service is a complex module capable of supporting structured queries for archive discovery. Two levels of security policies are enforced in this framework. At the application level the administrators can enforce global restrictions pertaining to the entire application and user community. At the group level, users creating new instances of collaborative groups can dictate group policies. Support for an extensible visual framework is provided. Moreover, users can subscribe to a set of notification topics and change their interface in a modular fashion at run-time without affecting other components.

The extensible OCGSA framework will simplify the development efforts of a large number of collaborative applications. The framework is in its initial design phase and is being submitted to the Global Grid Forum [10] for approval. At the same time efforts are being made on the implementation side to integrate the latest technologies into OCGSA, in particular the *Grid Service Flow Language* [11] to provide support for complex inter -service communication patterns. The core components will be implemented following the footsteps of OGSA [14]. A detailed formal draft of OCGSA specification is under development and its discussion is beyond the scope of this paper.

Acknowledgments

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; and by the NASA Information Power Grid program.

References

[1] Foster I., Kesselman C. and Tuecke S. (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3).

[2] Foster I., Kesselman C., Nick J. and Tuecke S. (2002) The physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf

[3] Christensen E., Curbera F., Meredith G. and Weerawarana S. (2000) Web Services Description Language. http://msdn.microsoft.com/xml/general/wsdl.asp

[4] Scheinblum, J. (2001) An introduction to SOAP. http://builder.cnet.com/webbuilding/0-7704-8-4874769-1.html

[5] von Laszewski, G., Pieper G. and Wagstrom P. (2002) Gestalt of the Grid. In Hariri S. and Parashar M. (eds), Performance Evaluation and Characterization of Parallel and Distributed Computing Tools, to be published.

[6] Open Grid Services Architecture Homepage. http://www.globus.org/ogsa

[7] Java Message Service Homepage http://java.sun.com/products/jms/

[9] The Jetspeed Webpage. http://jakarta.apache.org/jetspeed/

[10] Global Grid Forum Homepage http://www.gridforum.org

[11] Krishnan S., Wagstrom P. and von Laszewski G. (2002) GSFL: A Workflow Framework for Grid

Services. Preprint ANL/MCS-P980-0802

[13] Asbury B., Fox G., Haupt T., and Flurchick K. The Gateway Project: An Interoperable Problem Solving Environments Framework For High Performance Computing.

[14] OGSA Technology Preview 4: Homepage. http://www.globus.org/ogsa

[15] Globus Metacomputing Toolkit: Homepage http://www.globus.org

[16] Johnson C., Parker S., and Weinstein D. (1995) Large-scale computational science applications using the scirun problem solving environment. Proceedings of Supercomputing 1995

[17] Muralidhar R., Kaur S., and Parashar M (2000) An Architecture for Web-Based Interaction and Steering of Adaptive Parallel/Distributed Applications. Proceedings of Euro-Par 2000, pp ~ 1332-1339

[18] Access grid. Homepage http://www-fp.mcs.anl.gov/fl/accessgrid

[19] Agarwal D., Sachs S. and Johnston W. (1998) The Reality of Collaboratories. Computer Physics Communications vol. 110, issue 1-3. pp ~ 134-141

[20] Habanero Project homepage. http://www.isrl.uiuc.edu/isaac/Habanero/

[21] Groove Desktop Collaboration Software Homepage. http://www.groove.net/

[22] Nagaratnam, N, Janson P., Dayka J., Nadalin A., Siebenlist F., Welch V, Foster I. and Tuecke S. (2002) The Security Architecture for Open Grid Services. Version 1

[23] XML:DB Initiative for XML Databases: Homepage htp://www.xmldb.org

[24] Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S. (2002), A Community Authorization Service for Group Collaboration, proceedings of IEEE Workshop on Policies for Distributed Systems and Networks