Providing Portlet-Based Client Access to CIMA-Enabled Crystallographic Instruments, Sensors, and Data

Hao Yin[1, 2], Donald F. McMullen[3], Mehmet A. Nacar[1], Marlon Pierce[1], Kianosh Huffman[4], Geoffrey Fox[1], Yu Ma[5]

1) Community Grids Lab, The Pervasive Technology Labs at Indiana University, 501 N. Morton St. Bloomington, Indiana 47404

2) School of Computer Science, Sichuan University, No. 24, South Section 1, Yihuan Road, Chengdu, 610065, China

3) Knowledge Acquisition and Projection Lab, The Pervasive Technology Labs at Indiana University, 501 N. Morton St. Bloomington, Indiana 47404

4) School of Informatics, Indiana University, 901 E. 10th St., Bloomington, Indiana 47408

5) Department of Computer Science, Indiana University, 215 Lindley Hall, 150 S. Woodlawn, Bloomington, Indiana 47405

{hayin, mcmullen, mnacar, marpierc, kihuffma, gcf, yuma}@indiana.edu E-mail

Abstract

The Common Instrument Middleware Architecture (CIMA) project, supported by the NSF Middleware Initiative, aims at making scientific instruments and sensors remotely accessible by providing a general solution for services and user interfaces to remotely access data from instruments and to remotely monitor experiments. X-ray crystallography is one of several motivating applications for the development of CIMA. Data such as CCD frames and sensor readings may be accessed by portals through middleware services as they are being acquired or through persistent archives. CIMA software may be used to federate online instruments in multiple labs, so this project must also address problems in data management and data sharing. This paper describes a collaboration between the CIMA and the Open Grid Computing Environments projects (also supported by the NSF Middleware Initiative) to enable remote users to monitor instruments and interact with data gathered from CIMA-enabled crystallography laboratories through various Web portal components ("portlets") running within a standards-compliant portal container. We also discuss an approach taken to develop portlets that use Web Services for data management and solutions for managing distributed identity and access control.

1. Introduction

Remote access to shared instrument resources is a major outcome of e-Science development projects in many disciplines [1]. Shared remote access improves instrument utilization, collaboration between users and instrument experts and provides "hooks" for automating the processing of data coming from instruments by pre-configured workflows [2, 3].

One of the key issues in developing shared instrument systems is how to create an open and flexible approach to user interfaces for access to instruments and the data streams coming from them. In related work [4] we have described how portals can be used to organize access to instruments through the Common Instrument Middleware Architecture (CIMA) [5, 6] and how individual portlets can provide specialized, role and task specific functionality as users, technicians and system administrators interact in the generation, analysis and management of data from shared instrument resources. In this paper we will focus on the approach taken to develop portlets for managing crystallographic data in a group of cooperating laboratories.

2. Services for X-ray Crystallography

X-ray crystallography is an analytic technique to help scientists understand and determine the precise molecular structure of a crystalline substance. However the instruments (called X-ray diffractometers [7]) required to perform these types of studies are quite expensive and require a highly trained operator. The relevant data from a crystallography experiment contains a series of diffraction images usually captured by a CCD detector, and a number of environmental variables including crystal temperature, crystal alignment image, CCD cooling status, and the temperature and relative humidity of the lab.

In some cases, due to the nature of some crystalline materials such as proteins or microcrystalline compounds, the successful structure determination of these compounds require the use of high brilliance radiation sources available at national synchrotron facilities. Gaining access to beamlines at these national synchrotron facilities to collect data is not straight forward. Travel to these remote facilities is costly and time consuming, and once there, the facilities must be used in an intensive manner. By developing methodologies to remotely monitor and access instruments and their data we can provide the remote users with a "same as being there" experience with additional flexibility in scheduling around problem samples and equipment failures. Additionally on-site users and technicians can share data coming from the beamline's instruments with remotely located colleagues to discuss the quality of a diffraction pattern. This remote consultation capability can facilitate decision making such as continuing with a questionable sample or abandoning it and starting a new one. Effective shared access to instruments ensures a more efficient use of the beam time, potentially improving throughput of the beamline as a whole. This paper will focus on the implementation details of the CIMA crystallography portal and the mapping of end-user functional requirements to portlets.

3. The Common Instrument Middleware Architecture (CIMA)

Before discussing the CIMA portal architecture, we first review the services and instruments behind the scenes in order to motivate the portal requirements. Scientific instruments and sensors provide the raw observations used to develop, verify, and falsify theories. Data from instruments typically have an extensive lifecycle, which includes corrections and calibration, annotation, and then storage in a database or file system. The Common Instrument Middleware Architecture (CIMA) project [8], supported by the National Science Foundation Middleware Initiative [9], proposes a single virtualization layer to hide this complexity, and to present a relatively simple Web Service interface to the rest of the data pipeline. Further aims are to integrate instruments into software systems based on current cyber infrastructure standards, and to improve accessibility of instruments to both routine and non-routine users including software agents. CIMA middleware is based on current Grid implementation standards and supports a variety of instrument and controller types. CIMA implementations are being evaluated in three settings representing a spectrum of shared instrument applications: X-ray crystallography, remote robotic telescopes, and small sensor network nodes for environmental observation.

A brief description of CIMA crystallography application is given here below, and a more detailed description can be found in [2,5,6]. CIMA is in use in several crystallography labs in the US and abroad, including a synchrotron beamline, to acquire data from diffractometers and related sensors. Figure 1 shows a typical implementation combining data acquisition by CIMA with data management at a site remote from the lab where the instrument is located.

The CIMA component shown in Figure 1 as *Instrument Representative* (IR) can be embedded in the instrument or can be implemented in a proxy somewhere remote from the instrument but connected to it through an IP network or other interconnect bus. Interaction with a CIMA-enabled instrument is through the *CIMA channel* protocol using Web Services [5].

The IR streams real-time data over a CIMA *channel* to consumer applications that register for specific data and metadata provided by the IR. The main consumer application is a data manager (*My Manager)* that stores the data in a user defined location prior to reduction and analysis. *My Manager* provides data virtualization through the Obsidian data management library [10], which tracks the current physical location of data sets for reference by subsequent processing steps. It also maintains basic metadata about samples, runs and individual files that make up a sample run.

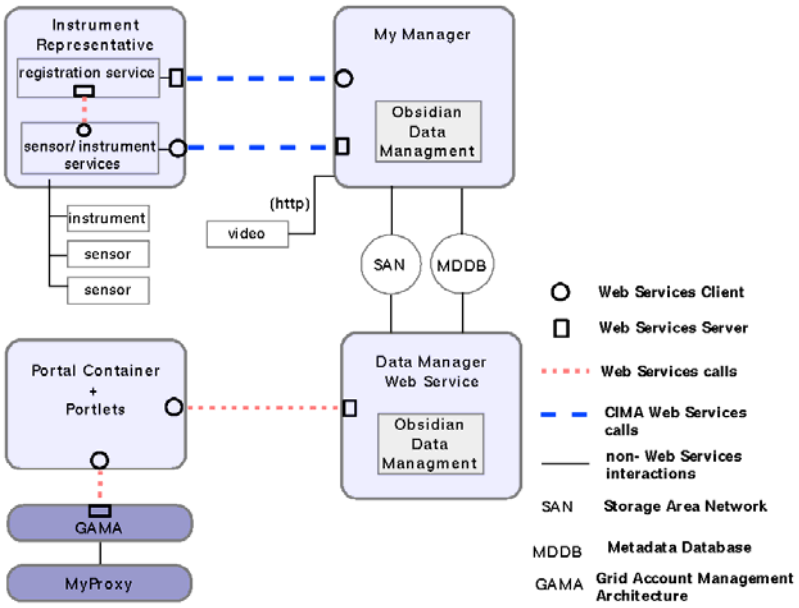

Figure 1. CIMA implementation (*Instrument Representative*) sends data from a laboratory to a remote data manager (*My Manager*) through Web Services based *CIMA channel*.

This paper is concerned with providing access to data through a portal interface using the *Data Manager Web Service* (DM-WS) in Figure 1, the portal client functions that must be supported include displaying data sets and individual components, *e.g.* CCD frames [11] from the diffractometer and their bitmap images, and other sensor readings for experiments in-progress or completed. Another important class of functions we must support through portlets is data stewardship tasks such as managing ownership and access rights to each sample collection. Future portlets will provide interfaces to analytical applications and archival functions.

4. Overview of Related Techniques

4.1 Data Portals and Portal Components

Web-based portals have been identified as a key enabling technology for e-research. Early efforts are extensively reviewed in Refs [12, 13] (see also [14, 15]) and considerable effort has been put into developing production quality portal systems for science. The key concept of a Web portal system is the ability for users to login to sites through browsers. Portals may provide publicly accessible pages for anonymous viewing, but by logging in and establishing identity, the portal can provide a number of additional features, such as access to restricted resources.

The core notion of science portals is that a research "community of interest" can provide ubiquitous access to many types of resources and can communicate and share knowledge through a common point of presence on the web. In practice portals provide community management functions such as identity management for individual participants, single sign-on for access to multiple communication, and compute and storage services needed by research communities. Individual users may be able to customize their view of the resources. ReciprocalNet [16] is an example of a portal for the sharing of chemical structures primarily determined by X-ray diffraction crystallography, and aimed at a diverse audience including researchers and educators.

Although portals may be built with any number of Web technologies, the large number of science portal activities (including the TeraGrid Science Gateway efforts [17] such as the LEAD portal [18]) means that the community benefits from adopting uniform, interoperable software. Here, we can make the distinction between portal containers, which provide common features such as login support and content customization, and their content components, known as portlets.

Standards have emerged for how portlets should communicate with their containing portal. JSR 168 [19] provides the key component definitions (programming interface and lifecycle) for Java-based portals. It has been widely adopted and is now

supported by a broad range of both commercial and open source portal projects such as GridSphere [20], uPortal, and Jetspeed2. The OGCE project [21] is actively developing and evaluating JSR 168 compliant portlets for e-research.

In early stages of the CIMA project (2004), a Jetspeed1 portal was used to display the sample data and various parameters via embedded CGI scripts in a portlet framework. In mid 2005 the CIMA portal project adopted the GridSphere portal container and started implementation of portlets to access instrument data. The portal currently serves hundreds of samples, some of which are available for viewing by the public.

GridSphere is an open source portal implementation compliant with JSR 168, which provides a portlet container and a collection of core services and portlets, such as login, logout, access control management and layout selection. GridSphere supports user and group management to take care of the authentication issue so that portlets are more flexible and easier to develop and maintain. We adopted GridSphere as our deployment environment based on its popularity in many of the science gateway projects, but the portlet components of our project are in principle portable to other containers.

4.2 JavaServer Faces for Portlet Development

Some of the limitations of the JSR 168 portlet specification are that it provides a limited development environment and does not provide reusable component widgets, so we need to inherit these from complementary development frameworks. Portlets may be developed from a number of technologies, including Struts, Velocity, JavaServer Pages, and JavaServer Faces through so-called portlet bridges. Three key requirements for this project are that portlets will contain various GUI widgets possibly beyond those in HTML forms, that the presentation widgets would be decoupled from the underlying data model they use, and that widgets be easy to test, preferably independently from portlets that contain them. JavaServer Faces (JSF) [22] has proven to be a good solution for meeting these requirements.

JSF is a specification for building user interfaces for server-side applications. One of the advantages is that JSF provides rich tag libraries to build components which run on the server and handle events generated by a client and can be rendered back to the client. The other advantage is that JSF is based on the Model-View-Controller (MVC) model [23], so it offers a clean separation between presentation and logic.

In addition to improving the flexibility with respect to back end data sources and reusability of portlets in general, GUI component beans developed for JSF applications can easily be unit tested and can reused in other types of applications.

4.3 Web Services

Web Services is a software system designed to support interoperable machine-to-machine interaction over a network [24]. Web Services interoperability and extensibility derive from the use of XML for framing messages and the use of Simple Object Access Protocol (SOAP) [25] to transport them between client and server. Web Services have been widely used to integrate applications in different languages on different platforms across organization boundaries. In CIMA data portal, portlets communicate with CIMA data management module (DM-WS) through Web Services. These Web services are Perl-based CGI scripts wrapped by SOAP::Lite for Perl [26], which is a collection of Perl modules providing a simple and lightweight interface to SOAP.

5. Implementation of the CIMA Crystallography Portal

5.1 Requirements

For the current work, a subset of requirements relating to user and administrative interaction with data was chosen. These include the following:

- Remote users and in-lab crystallographers must be able to monitor an experiment in progress, including viewing current and previously collected CCD frames and associated relevant environmental and technical parameters;
- All raw data is owned by the lab which performed the experiment and collected the data. In addition to the lab, represented by one or more lab administrators, individual users can view (but not modify or delete) their samples;
- Lab administrators must be able to control sample ownership and visibility;
- Because the notion of when an experiment ends is not clearly defined (*e.g.* experiments may be truncated after the fact or additional frames may be gathered based on evaluations made during a run), lab administrators should be able to set the end time of an experiment;
- Lab administrators must be able to add and remove users to an access control list for a sample;
- Users must be able to view their samples, including all files and sensor readings related to the experiment;

- Some sample data may be provided to the general public for educational or public science awareness purposes;
- Users must be able to view the current status of the lab as a whole;
- Individual functions that are of general utility should be implemented in a reusable, pluggable, standards-based manner as portlets that can be added or removed by administrators or end-users as appropriate;
- The portlets must interact with a lab's data manager software via Web Services calls;
- Users and groups will be managed by the portals container and access to all functions of the portal will be provided by a single sign-on through the portal.

A prototype implementation of the crystallography portal was completed using Jetspeed1 and CGI scripts. Although in the right direction, this implementation did not meet our modularization requirement and so with the ramp-up of the NSF middleware project and the availability of support from the Open Grid Computing Environments (OGCE) group, we migrated to GridSphere and JSF-based portlet clients to CIMA services as a fully JSR 168 compliant portal container. This assures a degree of survivability and lateral flexibility to move the science process specific functionality to other containers if the need arises.

The requirements outlined above led us to develop the following portlets:

- A lab overview portlet that provides the current status of a facility and its instruments;
- An administrative *Admin* portlet to support management of sample ownership and other parameters related to individual experiments;
- A *PublicSample* portlet that provides sample data to all portal users and the general public;
- A *UserSample* portlet that shows a logged-in user their samples and other samples on whose access control lists they appear.

PublicSample and *UserSample* portlets also allow users to scan through all data objects in an experiment (Figure 2). Per design choice some functions of the portlets are made available as pop-up windows. These portlets provide all of the functionality listed in the requirements above. Extensions to the portal's basic group authorization mechanism (discussed in Section 5.3) provide an access control list associated with each data collection. Scientists can view and modify sample data from their X-ray diffraction crystallography experiments based on their roles in this project and can add users to the access control lists of their data sets. Nothing more than a web browser is needed to interact with the system.

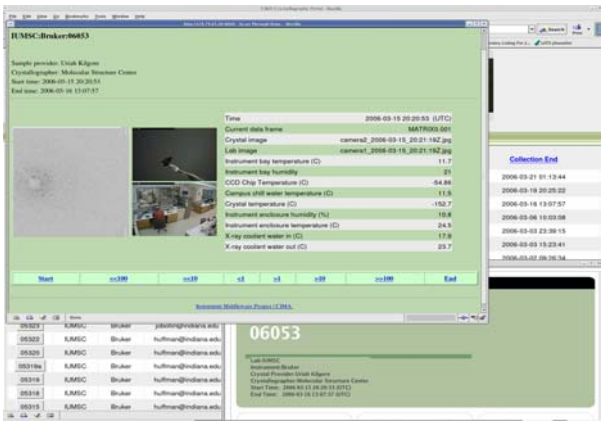

Figure 2. *UserSample* portlet that allows users to stepwise scan through an experiment.

5.2 Architecture of the CIMA Crystallography Portal

Developing JSF Web applications includes support for UI components, independent backing JavaBean code, and simplified management of HTTP request and session parameters. UI components handle the interaction with users and communicate with Web Services through managed beans such tasks as access to databases. Then we use portlet to wrap web application so that we can make use of the user, group and layout management from the portlet container (GridSphere), and also we can deploy these portlets with respective configuration for different laboratories within the same portal. Furthermore, the relationship between our portal and data manager is loosely coupled by using Web services, we can effortlessly deploy the portal and Web services at diverse locations.

JSF Web applications handle the UI events and navigation rules to implement the application controller logic. Stubs are generated according to the WSDL of Web services via WSDL2Java tool provided by Apache Axis [27]. This allows access to the Perl-based data manager services. The managed beans in the JSF Web application invoke on the stubs to communicate with Web services and store the data returned by Web services. JSF provides a value binding mechanism to make it easy for Web applications to represent the data combined with managed beans to users. Figure 3 shows the class diagram of some of these JavaBeans. The MVC structure of JSF allows us to concentrate on developing classes that model the instrument and lab environment.

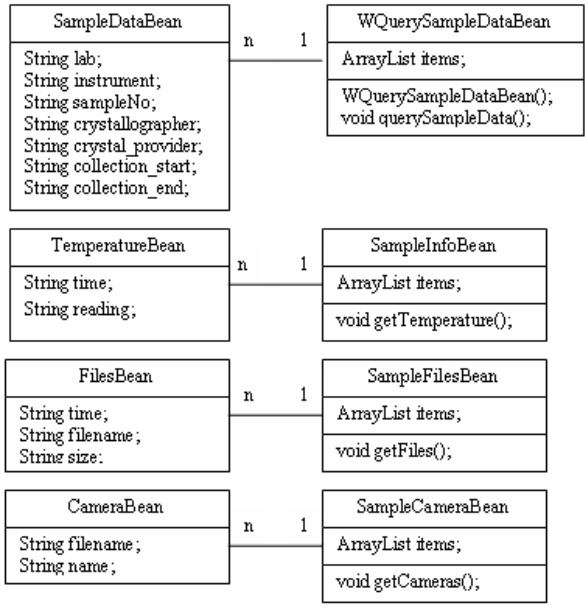

Figure 3. Class Diagram of JavaBeans

These beans are populated with information from Web Services calls to DM_WS. They are used to set up the model for sample data and related parameters. *WQuerySampleDataBean* and *SampleDataBean* acquire the basic information of sample data, such as sample number, laboratory, instrument and so on. *SampleInfoBean* and *TemperatureBean* obtain environmental conditions related to a specific sample, like temperature and humidity. *SampleFilesBean* and *FilesBean* inquire CCD frames both in raw and jpeg format pertaining to a specific sample. *SampleCameraBean* and *CameraBean* query camera images of laboratory and crystal during the experimental period.

In current CIMA crystallography portal, the JSF Portlet Integration Library [28] is used, which is built on the top of JavaServer Faces 1.1. This open source library deals with the low level communication between portlet and JSF: mapping portlet lifecycle onto JSF lifecycle; allowing JSF to interact with portlet APIs provided by containing environment, like GridSphere. It enables a seamless integration so that the CIMA JSF modules can be bundled with portlet to work inside GridSphere container.

5.3 Identity Mapping between Portal and Data Manager Service

A significant problem faced in the design of the CIMA crystallography portal is the mapping of identities and associated privileges of portal users to identities associated with data sets gathered by the *My Manager* component.

The authorization model used by the portal container is that of users assigned to a limited set of four roles (Super, Admin, User, Guest) whereas the data manager uses an authorization model with users, groups and access control lists that can contain users and groups. Gridsphere does have a notion of groups but this is related to what users can access which portlets rather than a Unix-like notion of a general authorization mechanism for sets of users. Since the portal does not provide a flexible authorization scheme a design choice was made to perform the mapping of portal users to data owners, groups and access control lists in the logic of the portlets used to access the data manager.

As mentioned above, there are three portlets, *PublicSample* portlet, *UserSample* portlet and *Admin* portlet. For *UserSample* and *Admin* portlets, users can only access the related data according to their roles. Thus, an approach is required to match user identities between portal and data manager service.

A logged-in GridSphere user's name can be obtained through the JSF context via the portlet-based external context supplied by [28]. The following is a fragment of such codes:

```
FacesContext cxt = FacesContext.getCurrentInstance();
HttpServletRequest request =
    (HttpServletRequest)cxt.getExternalContext().getRequest();
Map userInfo =
    (Map)request.getAttribute(PortletRequest.USER_INFO);
String username = (String) userInfo.get("user.name");
```

Then the *username* can be used to query the database combined with GridSphere to get the information of the user, such as user's full name, email address and groups, etc. Because the current version of GridSphere doesn't support a hierarchy of groups containing users, we use GridSphere portal groups to control access to portlets that, in turn, control access to data objects. Several GridSphere portal groups are created: *"<lab>_admin"* represents the lab administrators who can access *Admin* portlet and are in charge of the ownership and access control of samples; *"<lab>_client"* represents portal users who are clients from a single lab and can access *UserSample* portlet. Access control lists are implemented as other groups titled as research group names, the members of which can view sample data from the group they belong to (non-public samples). The lab administrator can control the access list for a sample through *Admin* portlet according to users and groups mentioned above. The first two groups (*<lab>_admin* and

<lab>_client) are used to separate users from different laboratories when there are multiple CIMA crystallography portals deployed in the same container. Gridsphere's SPORTLETUSERIMPL, SPORTLETGROUP and GROUPREQUESTIMPL database tables are required to get these information through JDBC [29] programming.

Finally, the user name and groups information are transferred as parameters to Web Services to fetch sample data.

6. Conclusion and Future Work

The work described here provides a model for organizing and accessing data from CIMA instruments and identifies a problem and provisional solution for managing identity relationships more complex than those provided by GridSphere.

The framework of CIMA crystallography portal is based on the foundation of extensively vetted standards, JavaServer Faces, JSR 168 Portlet and Web services. It is easy and flexible for development, deployment and maintenance.

More importantly, the difficulties encountered in this project with authorization indicate that more work is needed to provide a flexible, general solution for authentication and authorization within and between portal instances. As described above, we have taken a pragmatic approach for solving a specific problem (merging portal and data service identity). However, our system is a good candidate for investigating Web Service security standards [30, 31] for securely communicating authentication tokens and access privileges. See [32] for related work in this area.

Another interesting problem is the integration and federation of multiple instances of the CIMA portal. CIMA collaborates with international research groups, some of which are setting up their own CIMA portal instances. The Web Services for Remote Portlets (WSRP) specification [33] (currently under revision) is one interesting possibility for future investigation. Its integration with Web Service security standards will be an interesting challenge.

7. Acknowledgements

CIMA was supported by National Science Foundation cooperative agreements and grants SCI 0330568 and MRI CDA-0116050, respectively. OGCE development is supported by the National Science Foundation's Middleware Initiative, SCI 0330613.

We thank Professor Jiliu Zhou, School of Computer Science, Sichuan University, China for supporting H.Y.; Professor Randall Bramley and Tharaka Devadithya, IU Computer Science Department for contributions to CIMA design and implementation; Dr. John C. Huffman, IU Chemistry Department, and all the crystallographers at the various CIMA participating laboratories worldwide for their invaluable help and feedback; and Ji Young Chong of the IU Department of Telecommunications for the graphical design of the CIMA crystallography portal.

8. References

[1] Remote Instrumentation, http://science.internet2.edu/remote.html.

[2] R. Bramley, K. Chiu, T. Devadithya, N. Gupta, C. Hart, J. C. Huffman, K. Huffman, Y. Ma, D. F. McMullen, "Instrument Monitoring, Data Sharing, and Archiving Using Common Instrument Middleware Architecture (CIMA)", *J. Chem. Inf. Model.,* 2006. DOI: http://dx.doi.org/10.1021/ci050368l

[3] S. J. Coles, J. G. Frey, M. B. Hursthouse, M. E. Light, K. E. Meacham, D. J. Marvin and M. Surridge, "ECSES – examining crystal structures using 'e-science': a demonstrator employing web and grid services to enhance user participation in crystallographic experiments", *J. Appl. Cryst*. 2005, 38, pp. 819-826.

[4] D. F. McMullen, K. Huffman, "Connecting Users to Instruments and Sensors: Portals as Multi-user GUIs for Instrument and Sensor Facilities", In Proceedings of GCE 2005: Workshop on Grid Computing Portals held in conjunction with SC05, Seattle, WA, Nov. 18, 2005. Submitted to the *Journal of Concurrency and Computation: Practice and Experience*.

[5] T. Devadithya, K. Chiu, K. Huffman, D.F. McMullen, "The Common Instrument Middleware Architecture: Overview of Goals and Implementation", In *Proceedings of the First IEEE International Conference on e-Science and Grid Computing (e-Science 2005)*, Melbourne, Australia, Dec. 5-8, 2005. http://doi.ieeecomputersociety.org/10.1109/E-SCIENCE.2005.77

[6] D.F. McMullen, T. Devadithya, and K. Chiu, "Integrating Instruments and Sensors into the Grid with CIMA Web Services", In *Proceedings of the Third APAC Conference and Exhibition on Advanced Computing*, The APAC Conference and Exhibition on Advanced Computing, Grid Applications and e-Research (APAC05), Gold Coast, Australia, Sept 26-30, 2005. http://www.vpac.org/ocs/viewpaper.php?id=55&cf=3

[7] C. Giacovazzo, *Fundamentals of Crystallography*, Oxford University Press, New York, 1992.

[8] The Instrument Middleware Project, Available http://www.instrument-middleware.org

[9] NSF Middleware Initiative, Available http://www.nsf-middleware.org

[10] Y. Ma, R. Bramley, "Obsidian, A Composable Data Management Architecture for Scientific Applications", In Proceedings of Challenges of Large Applications in Distributed Environments, Research Triangle, NC, 2005.

[11] S. W. Muchmore, "Experiences with CCD detectors on a home X-ray source", *Acta Cryst.* 1999, D55, pp. 1669-1671.

[12] F. Berman, G. Fox, T. Hey, (eds.). *Grid Computing: Making the Global Infrastructure a Reality,* John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0 (2003). http://www.grid2002.org.

[13] A. Hey, G. Fox, (eds.) *Concurrency and Computation: Practice and Experience*, Vol. 14, No. 13-15 (2002). Special issue on Grid Computing Environments.

[14] D.E. Atkins, K. K. Droegemeier, S. I. Feldman, H. Garcia-Molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker, and M. H. Wright, "Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure", Technical Report, January 2003.

[15] D. Gannon, G. Fox, M. Pierce, B. Plale, G. von Laszewski, C. Severance, J. Hardin, J. Alameda, M. Thomas, and J. Boisseau, "Grid Portals: A Scientist's Access Point for Grid Services (DRAFT 1)", GGF working draft, Sept. 19 2003. Available ww.computingportals.org/meetings/next_mtg.php/docs/GCE-Portal-working-draft.pdf

[16] L. M. Sandvoss, W. S. Harwood, A. Korkmaz, J. C. Bollinger, J. C. Huffman, and J. N. Huffman, "Common Molecules: Bringing Research and Teaching Together Through an Online Collection", *Journal of Science Education and Technology*, Volume 12, Issue 3, Sep 2003, pp. 277 – 284.

[17] TeraGrid Science Gateways Program: http://www.teragrid.org/programs/sci_gateways/

[18] B. Plale, D. Gannon, D. A. Reed, S. J. Graves, K. Droegemeier, B. Wilhelmson, M. Ramamurthy, "Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD", International Conference on Computational Science (2) 2005, pp. 624-631.

[19] A. Abdelnur, S. Hepper, "Java Portlet Specification version 1.0", 2003, Available http://www.jcp.org/aboutJava/communityprocess/final/jsr168

[20] J. Novotny, M. Russell, O. Wehrens, "GridSphere: a portal framework for building collaborations", *Concurrency - Practice and Experience*, 2004, 16(5), pp. 503-513.

[21] J. Alameda, M. Christie, G. Fox, J. Futrelle, D. Gannon, M. Hategan, G. Kandaswamy, G. von Laszewski, M. A. Nacar, M. Pierce, E. Roberts, C. Severance, M. Thomas, "The Open Grid Computing Environments Collaboration: Portlets and Services for Science Gateways." Accepted for publication in "Science Gateways" special issue of Concurrency and Computation: Practice and Experience.

[22] C. McClanahan, E. Burns, and R. Kitain, Eds., "Java Server Faces Specification version 1.1", 2004, Available http://java.sun.com/javaee/javaserverfaces

[23] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*, John Wiley & Sons, 1996.

[24] W3C Working Group, "Web Services Architecture", W3C, 2004, Available http://www.w3.org/TR/ws-arch

[25] Simple Object Access Protocol. http://www.w3.org/TR/soap

[26] P. Kulchenko, SOAP Lite project, Available http://www.soaplite.com

[27] WebServices-Axis, Available http://ws.apache.org/axis

[28] JavaServer Faces Portlet Integration Library, Available https://javaserverfaces.dev.java.net/servlets/ProjectDocumentList?folderID=3389&expandFolder=3389&folderID=1504.

[29] JDBC Technology, Available http://java.sun.com/products/jdbc

[30] OASIS Web Services Security TC, "WS-Security", OASIS, Available http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[31] OASIS Security Services TC, "Security Assertion Markup Language (SAML)", OASIS, Available http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[32] L. Fang, D. Gannon, F. Siebenlist, "XPOLA: An Extensible Fine-grained Authorization Infrastructure for Web Services in Grids", In *PKI R&D 05, April, 2005.*

[33] A. Kropp, C. Leue, R. Thompson (eds), "Web Services for Remote Portlets Specification 1.0" Approved as an Oasis Standard August 2003.