JSR 168 AND WSRP 1.0 – HOW MATURE ARE PORTAL STANDARDS?

Xiaobo Yang, Xiao Dong Wang, Rob Allan
CCLRC e-Science Centre, Daresbury Laboratory, Warrington WA4 4AD, UK

Keywords: Portal, Portlet, JSR 168, WSRP, Grid, e-Science.

Abstract: The benefits of adopting web portals in different scenarios like e-Learning and e-Research are well understood now. With built-in single sign-on (SSO), role-based authorisation management and personalisation, portals provide a uniform interface for seamless access by users to existing or emerging distributed resources such as the Grid. In this paper, two portal standards – Web Services for Remote Portlets (WSRP) and Java Specification Request (JSR) 168 will be discussed to reveal how practical they are in developing real world portals. The discussion is based on our work in portal development for several UK e-Science projects including the UK NGS (National Grid Services) Portal and the Sakai VRE Portal Demonstrator project.

1 INTRODUCTION

Web portals can play a prominent role in real world applications by bridging end-users to resources and hiding underlying middleware complexity. Resources could be as simple as some arbitrary data stored in a database, or as complex as business logic on the Grid. By aggregating distributed resources together with built-in SSO, role-based authorisation and support for personalisation, portals can streamline the use of distributed resources and improve the productivity of existing software systems.

With the widening use of Grid technology, portals are often used to provide transparent client access. Prior to the two portal standards - WSRP 1.0 (WSRP1.0) and JSR 168 (JSR168) born in 2003, portals were developed with a lot of similar code rewritten in different Grid/e-Science projects. Toolkits like GPDK (Novotny, 2002), Java CoG (von Laszewski, 2001) and GridPort (Thomas, 2001) were developed to simplify the task. This approach also led to many customised libraries created to meet the demands of particular projects, such as the Grid Application Toolkit (GAT). Although these well defined APIs/libraries can help to simplify portal development, non-standard based portal applications (Peltier, 2003, Wu, 2004, Bondarescu, 2005) are not easy to re-use outside of the original project.

Realising the importance of standards in portal development for the UK e-Science community, an international workshop titled *Portals and Portlets* was held in July 2003 at the UK National e-Science Centre. It covered the main portal work of the Grid community at that time. Work of GridSphere, NEESgrid, OGCE, uPortal together with IBM WebSphere Portal, Sun One Portal Framework and more projects were presented. Shortly after that workshop, WSRP and JSR 168 were formally ratified by OASIS (Organisation for the Advancement of Structured Information Standards) and JCP (Java Community Process) respectively to solve the interoperability issues in portal development. At the beginning of March 2005, another workshop on *GridSphere and Portlets*, was held with discussions focusing on sharing JSR 168 portlets between different frameworks, plus an initial investigation of WSRP.

WSRP and JSR 168 are slowly becoming adopted by portal vendors and developers. Today plenty of open-source and commercial portal frameworks are available on the market, for example, eXo platform, Liferay, uPortal, JBoss Portal and IBM WebSphere Portal. They all claim to support JSR 168 and many also claim to support WSRP.

In this paper, we will first give an introduction to WSRP and JSR 168. Then explain lessons learnt from development of the NGS Portal. In this section,

Yang X., Dong Wang X. and Allan R. (2006).
JSR 168 AND WSRP 1.0 – HOW MATURE ARE PORTAL STANDARDS?.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 393-399
DOI: 10.5220/0001252503930399

re-use of the business logic and presentation layer will be discussed in the context of a test of standard compliance of selected open-source portal frameworks. Some related work will be discussed before giving concluding remarks and an outline of possible future work.

2 TWO PORTAL STANDARDS

2.1 JSR 168

Many on-line resources such as IBM's *DeveloperWorks*, are available providing introductory and in-depth discussions of the JSR 168 standard. Here we give a brief introduction for completeness. JSR 168, also called the Java Portlet Specification 1.0, is designed to standardise the interaction between portlet and portlet container (portal framework) by the Java Community Process. In JSR 168, **portal**, **portlet** and **portlet container** are defined as follows.

Portal - *"A portal is a web based application that – commonly – provides personalisation, single sign on, content aggregation from different sources and hosts the presentation layer of information systems."*

Portlet – *"A portlet is a Java technology based web component, managed by a portlet container, that processes requests and generates dynamic content.*

Portlet Container – *"A portlet container manages portlets and provides them with the required runtime environment."*

User requests are therefore managed by a portlet container and transferred to its portlets. A portal is an aggregated view of the dynamic content generated by several portlets. Although a portlet container can be built as a separate component in a portal application, it is commonly integrated with the portal to become a fully functional portal framework. Whilst the portlet container is focussed on managing the life cycle and request process of portlets deployed inside it, the portal normally provides extra functionalities such as SSO, role-based authorisation support and personalisation in addition to rendering to provide a consistent "look and feel". Because of the prevalent integration of portlet container and portal, "portlet container" and "portal framework" are both commonly utilised to describe the combined functionality.

Fig. 1 illustrates the relationships between end-user, portal, portlet container and portlets. In Fig. 1, the indicated business logic could be either inside or outside the portlet.

JSR 168, as its name implies, only appropriate for the Java programming language. This brings the issue – how to re-use web contents published using languages other than Java, for example, Perl or C CGI and PHP? Also, a definition for exchanging information between portal frameworks, e.g., re-use of remote portlets, is missing in JSR 168. The WSRP specification was developed to meet these requirements.

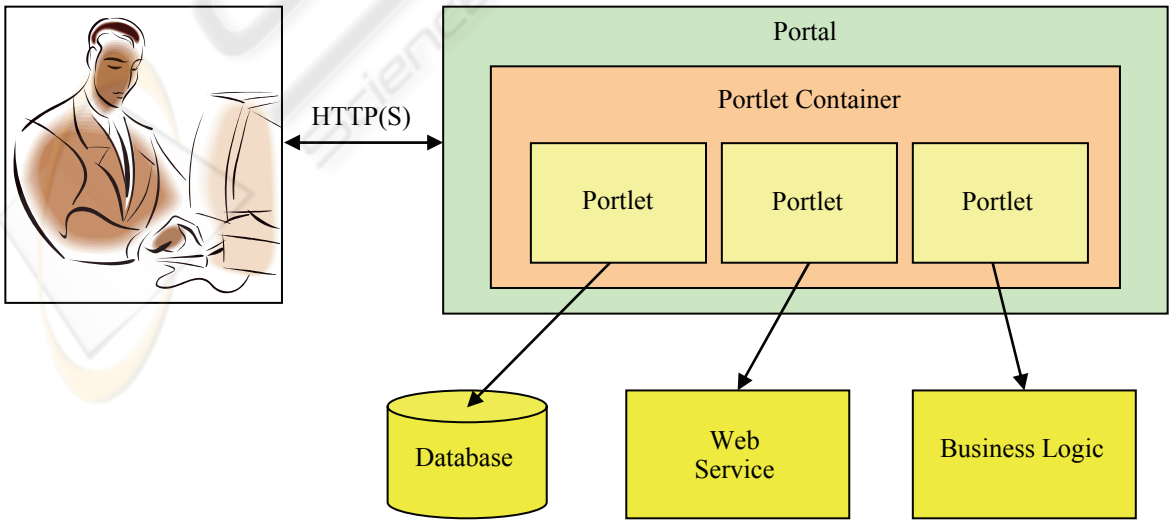

Figure 1: JSR 168 defines standard between portlets and portlet container.

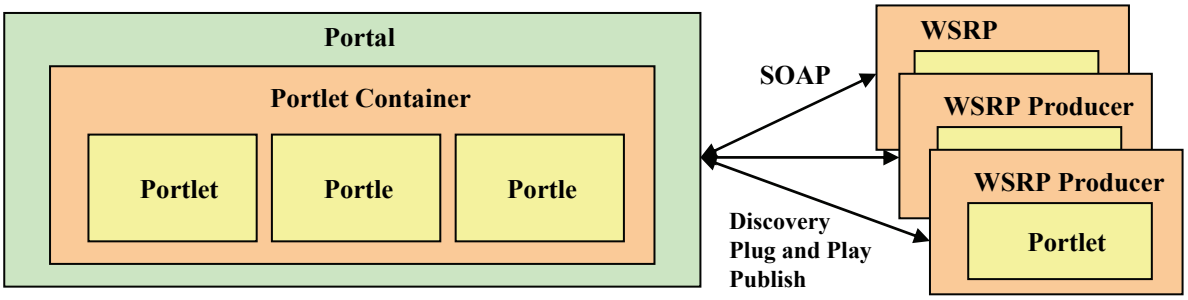

Figure 2: WSRP 1.0 defines contract for portlet containers to exchange information.

2.2 WSRP 1.0

WSRP 1.0, an approved OASIS standard, was defined as *"a web services interface for accessing and interacting with interactive presentation-oriented web services"*. Unlike JSR 168, WSRP is based on the web service concept; itself is based on language- and platform-independent technologies like SOAP, WSDL and UDDI. Therefore in theory, it is possible to use programming languages other than Java to provide information (defined as *Producer* in WSRP 1.0) which can then be consumed by any type of clients (defined as *Consumer* in WSRP 1.0) although normally a *Consumer* is a web portal. Unfortunately until now there are few implementations of WSRP producers using languages other than Java. The only one known to the authors is the Go-Geo! portal from EDINA (Awre, 2005, Go-Geo!). This is a Perl application using the SOAP::Lite web services module.

To explain the lack of other language implementation of WSRP, we note that the WSRP 1.0 specification (Section 1.2.2 line 25) suggests *"Producers are modelled as containers of Portlets"*. WSRP4J, a well-known WSRP Java implementation makes use of Pluto, a reference implementation of JSR 168 as the portlet container. As there is no portlet standard in other languages, this makes it much more difficult to implement the WSRP specification. From this point of view, WSRP 1.0 is actually highly coupled with the JSR 168. Alternatives to Java are however available to include remote web sites into a portal for example through the "IFRAM" tag. Furthermore, in its SharePoint Server 2003 (SharePoint), Microsoft announced both WSRP Producer and Consumer support through WSRP Web Services Toolkit and WSRP Web Part Toolkit respectively. This makes it possible for third-party portals to leverage SharePoint application functionalities as well as to consume WSRP portlet services provided by a variety of vendors, regardless of the business system used.

Fig. 2 illustrates the relationship between a portal equipped with a WSRP Consumer and some WSRP Producers. A portal can be constructed using local and remote portlets.

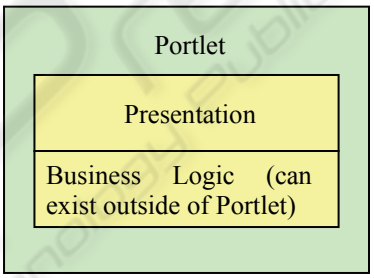

Figure 3: Portlet contains both business logic and presentation layer.

As illustrated in Fig. 3, a portlet acts as a web component which has integrated presentation and business logic. The latter can be external services with which the portlet can communicate. The key idea of WSRP is to re-use both the logic and presentation layers with the help of the prevalent web service concept. As mentioned above, WSRP 1.0 defines *Producer* and *Consumer* to a simulate web service and its client. As stated in the specification, WSRP is a *"protocol [which] describes conversation between Producers and Consumers on behalf of End-Users (clients of the Consumer)"*. The WSRP Producer also acts as a portlet container and portlets inside it are only accessible via the Producer, the portlets are not web services themselves. The four interfaces defined by the WSRP 1.0 specification are now described:

1) **ServiceDescription interface** – this required interface enables a consumer to discover the services that a producer provides by defining an operation for acquiring the producer's metadata;

2) **Markup interface** – the second required interface, is used to define operations for getting the

markup from a portlet and process user interactions with that markup. It also handles HTTP cookies;

3) **Registration interface** – an optional interface which enables a consumer to register at the producer by defining operations for establishing, updating and destroying a registration;

4) **PortletManagement interface** – another optional interface which covers lifecycle and properties of portlets. It defines operations for getting portlet metadata, cloning portlets for further customisation and interacting with the property interface.

A WSRP Consumer gives a portal the capability to render portlets maintained remotely. A portal page may present both local and remote portlets in a way transparent to end-users and even portal administrators (see Fig. 2). For example, in StringBeans 3.0 the portal will try to create an instance of a WSRP proxy portlet for each available remote portlet during its startup phase. All remote portlets are then treated as if they were local and are added to the list of those available.

Similar to JSR 168, there are introductory materials about WSRP 1.0 available online (for example, Gupta, 2005) but almost all of them are limited to an overview of the specification itself without any real programming details. We will discuss below experiences from our practical work on WSRP.

3 EXPERIENCES OF PORTAL DEVELOPMENT

3.1 Re-Use of Business Logic via JSR 168 Portlets

A web portal for the UK National Grid Services (NGS), the NGS Portal (Yang, 2005A), was developed in the Grid Technology Group at the CCLRC Daresbury Laboratory. As JSR 168 was chosen as the most appropriate technology for this work, the NGS Portal release 2.0 is based on a customised version of StringBeans. A MyProxyLoginModule using JAAS was added to StringBeans in order to help authenticating users through the NGS MyProxy server directly without needing to have a local account pre-defined in the portal. A set of JSR 168 portlets has been converted from the NGS Portal release 1.0 (which used Jetspeed 1.0 portlets deployed inside CHEF, a CompreHensive collaborativE Framework now replaced by the Sakai project (Sakai)). The portlets listed below use the JavaCoG to manage Globus Toolkit 2 (GT2) functionality from the web.

- ProxyManager portlet
- JobSubmission portlet
- BatchJobMonitor portlet
- FileTransfer (GridFTP/SRB) portlet
- LDAP/MDSBrowser portlet

These portlets were initially developed under the eXo platform and then ported to StringBeans for the production portal. During development, they were also tested in GridSphere and uPortal to check interoperability. It was proven that the JSR 168 standard solves the interoperability issues quite well and portlets can be re-used in different portlet containers. There is no need to modify the portlet source code, only some minor modifications of the configuration file (web.xml) and replacement of tag libraries. A portal framework like GridSphere needs more configuration files than the standard portlet.xml and web.xml. Some of the portlets designed for the NGS Portal were successfully used in GridSphere to clone a prototype portal for the Integrative Biology project (IB Project). Fig. 4 gives a screenshot of the BatchJobMonitor portlet.

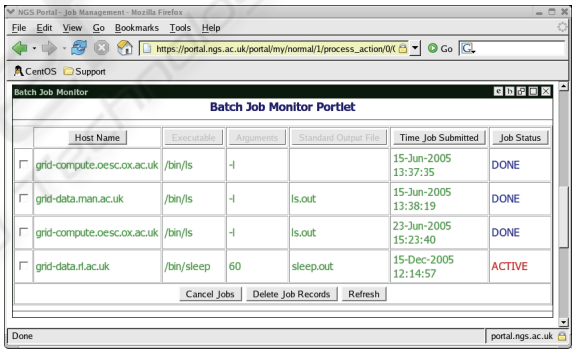

Figure 4: A screenshot of the NGS Portal – BatchJobMonitor portlet.

Our experience on eXo platform, GridSphere, StringBeans and uPortal shows that portlets can be simply re-utilised with only some minor modifications of several configuration files. The JSR 168 standard is quite mature on the market today. Obviously, different portal frameworks provide their own functionalities extending the standard JSR 168 specification, but this is not guaranteed to be portable. In practice, we therefore recommend to stay with standard functions provided by JSR 168.

3.2 Re-Use of Business Logic and Presentation via WSRP

After the NGS Portal 2.0 was released, we investigated various more complex aspects of portal/ portlet development. Part of the work involved a test of WSRP support in selected open-source portal frameworks. This includes eXo platform, Liferay, StringBeans, uPortal and WSRP4J. Although re-use of business logic in our scenario through portlets is the most common case, a further step can be achieved by re-using remotely maintained portlets which also contain a presentation layer. This approach eliminates portlet re-deployment and makes it potentially quick and easy to set up a new portal just by linking to WSRP Producers.

Despite the widespread claims, our tests showed that WSRP support is however still immature (Yang, 2005B).

Till now, no universal Consumer exists that can access all types of WSRP Producers. Even though the WSRP 1.0 specification has been available for two years, interoperability between portal frameworks is still poor. Issues for both WSRP Producer and Consume are listed below:

1) The WSRP 1.0 specification defined four interfaces but only two of them – *ServiceDescription* and *Markup* interfaces are mandatory. The other two interfaces – *Registration* and *PortletManagement* are optional, but these two optional interfaces play an important role in registration and remote portlet lifecycle management.

2) In the specification, it is mentioned that two forms of registration are supported:

- In-band registration – this requires that the Consumer sends a request to register with the Producer;
- Out-of-band registration – the Producer and Consumer go through specific business processes to establish registration.

It is clear that both registration methods require the optional *Registration* interface while at the same time the out-of-band registration requires further semantics and a process to be agreed for communications between a Consumer and a Producer.

3) Three URL types – **blockingAction**, **render** and **resource** are defined in the WSRP 1.0 specification, but it was observed that this is not well implemented on either Producer- or Consumer-side. For instance, under the circumstance of Consumer-side URL re-writing a Producer should indicate a static image in the markup as a **resource** URL type. Then the corresponding Consumer should re-write the URL to point to the correct location. But if the remote portlet does not encode such an image URL, then none of the WSRP Producers/Consumers we have tested can display the image correctly while encoding of such a resource URL is not necessary.

Such statements make it very difficult to write a universal WSRP Consumer to handle all situations. For this reason it can be observed that each portal framework's Consumer works best with its own Producer. There are other issues that the WSRP 1.0 specification does not cover. For example, if registration information of a Producer is changed there is no mechanism to notify its Consumers. This may be necessary since the Producer could ask a registered Consumer to re-register.

Currently we are working on a servlet-based WSRP Consumer which internally accesses a UDDI registry. End-users can search the registry to get a list of available remote portlets that meet their criteria (currently keyword based). The remote portlet can then be selected and run on behalf of the user. Our initial work has shown some promising results. Fig. 5 gives a screen shot of our WSRP Consumer servlet running inside uPortal through its Inline Frame function - a *Hello World* Portlet is up and running. The next step will involve some further work on our WSRP Consumer and it is planned to port to the Sakai framework as part of a Sakai VRE Demonstrator project. Third-party remote portlets such as the portlets developed for the NGS Portal could then be invoked alongside the Sakai collaboration and educational tools.

Figure 5: Integrating MyWSRPConsumer servlet inside uPortal through its Inline Frame function.

Similar to WSRP 1.0, JSR 168 also does not solve all the issues we identified. Some of them are listed below:

1) Lack of inter-portlet communication is always noted by portlet developers (Osmond, 2005). Some portal framework vendors have their own solutions

which extend the specification, like IBM in its WebSphere portal.

2) Support for different web technologies like Struts and JSF is not always available or complete, although portlet container providers solve this issue by providing different bridges. For example, Portals Bridges used in Jetspeed provides support for JSR 168 compliant development using common web frameworks like Struts, JSF, PHP, Perl and Velocity. Some similar bridges have also been integrated by other portal framework vendors including JBoss Portal, GridSphere, StringBeans and Vignette Portal.

3) Lack of portlet filter to add processes before and after accessing portlets. The Apache portals project provides this kind of function which is useful, e.g. for validating or modifying requests and responses.

Issues listed above need to be addressed by standards to improve the portability of portlets and avoid the need for framework-specific extensions. This is the basis of the work now being done by the portal community to develop JSR 286 (Portlet Specification 2.0) and WSRP specification 2.0.

4 RELATED WORK

Grid portals first emerged just after the introduction of the Grid concepts in the mid 1990s. User friendly interfaces were required for seamlessly accessing integratied distributed Grid resources. Portals were a natural choice for this because of the prevalence of web-based applications and extended the simple use of a browser to view and download information. As described earlier, the first-generation portals were normally project-based, but led to middleware toolkits like GridPort and JavaCoG. The two portal standards, WSRP 1.0 and JSR 168, make it easier to re-use generic functionalities and portals today are widely adopted by different domains beyond the Grid, including e-Science, e-Learning and e-Research. The second-generation web portals are based on these portal standards particularly on JSR 168. Some of them were based on an underlying Service-Oriented Architecture (SOA).

NEESgrid links earthquake researchers across the USA with leading-edge computing resources and experimental research equipment. Through NEEScentral, a web-based portal, all NEES participants and earthquake engineering researchers can make use of community-wide tools and resources integrated within NEESgrid in collaborative project areas.

The GEON (Geoscience Network) cyber-infrastructure project which involves a number of institutions and industry partners in USA is based on SOA. GEON aims at creating an IT infrastructure to enable interdisciplinary geoscience research. Besides the web service interface, GEON has a portal for end-users to access a set of portlets from which services like the visualisation tool.

The GroupLog and CREE (Contextual Resource Evaluation Environment) projects investigated creating web portals as the presentation layer for a variety of information search tools. Both projects evaluated JSR 168 and WSRP during their development. JSR 168 compliant portlets have been proven to work well within their portal environments, but for WSRP only CREE (Awre, 2005) has WSRP4J plus uPortal tested and reported to work well. Within the GroupLog project (Duke, 2005) a JSR 168 portlet has been written to call Perl CGI scripts as business logic, illustrating a candidate way to “glue” non-Java legacy applications into portal technology.

Finally we note the work done by the EDINA group in developing the Go-Geo! Portal which featured a WSRP producer written in Perl. Clearly this could also be done in other languages which support web services.

5 CONCLUSIONS AND FUTURE WORK

Web portals today play an important role in real world applications because they are bridging end users to resources behind the web. With the help of portals, distributed resources and data can be accessed as Grid or web services can be seamlessly aggregated with built-in SSO and authorisation support, personalisation support and much more.

JSR 168 and WSRP 1.0 are the two currently standards used in portal development. Both standards are designed to enable interoperability and re-use, with JSR 168 focusing on interoperability between portlets and portlet containers and WSRP 1.0 between portlet containers. Today all major portal frameworks claim JSR 168 support and quite a few of them also claim WSRP support. It has been observed that JSR 168 is very well supported in all frameworks tested but WSRP support is still immature. We currently suggested adopting JSR 168 for portal development and adding WSRP support later when it becomes more mature – the standards are essentially independent. Using these standards should mean that portlet developers do not need to worry about re-use of their portlets which should be guaranteed by portal framework vendors. Current

JSR 168 compliant portlet containers can plug in a WSRP Producer support module and they can provide a Consumer like the ProxyPortlet in uPortal as a generic WSRP Consumer. StringBeans 3.0 adopted this approach for its recently announced WSRP support.

As described above, both specifications have their own issues to deal with in the future, for instance, lack of inter-portlet communication in JSR 168 and some uncertainties (for example, complexity) in WSRP 1.0. Although still not available to test, their successors JSR 286 and WSRP 2.0 are now expected to solve some of these kinds of issues which will make future portal/ portlet development much easier.

Finally we note that the Sakai project has recently been very active in developing its own kernel WSRP Producer. As it is crucial to our Sakai VRE Portal Demonstrator project, further investigation will be carried to study the possibility of integrating uPortal and Sakai, which will lead to the maximum re-use of existing resources. In the further, we will also study security issues around remote portlets.

REFERENCES

Awre, C., Waller, S., Allen, J., Dovey, M.J., Hunter, J., and Dolphin, I. 2005. Putting the library into the institution: using JSR 168 and WSRP to enable search within portal frameworks. *Ariadne*, 45, from http://www.ariadne.ac.uk/issue45/awre/.

Bondarescu, R., Allen, G., Daues, G., Kelley, I., Russel, M., Seidel, E., Shalf, J., and Tobias, M. 2005. The astrophysics simulation collaboratory portal: a framework for effective distributed research. *Future Generation Computer Systems*, 21, 259-270.

Duke, M., and Swift, E. 2005. Portlet Feasibility Study: a report prepared for the GroupLog project funded by JISC under the eTools programme. From http://www.bath.ac.uk/e-learning/grouplog/jisc/ groupLog-portlet-feasibility.pdf.

Gupta, R.K. 2005. WSRP: Dynamic and real-time integration: an introduction to WSRP, its usage, and implementation. *WebServices Journal*, 5(8), 10-19, from:http://webservices.sys-con.com/read/121937.htm

IB Project. http://www.integrativebiology.ox.ac.uk/.

Go-Geo!. http://hds.essex.ac.uk/Go-Geo/.

JSR168. JSR-168 portlet specification. http://www.jcp. org/aboutJava/communityprocess/final/jsr168/.

Novotny, J. 2002. The Grid portal development kit. *Concurrency Computat: Pract. Exper*., 14, 1129-1144.

Osmond, M., and Guo, Y. 2005. Adopting and extending portlet technologies for e-Science workflow deployment. In *UK e-Science AHM 2005, Nottingham, UK*, available on CDROM.

Peltier, S.T., Lin, A.W., Lee, D., Mock, S., Lamont, S., Molina, T., Wong, M., Dai, L., Martone, M.E., and Ellisman, M.H. 2003. The telescience portal for advanced tomography applications. *J. Parallel Distrib. Comput*., 63, 539-550.Sakai. http://www.sakaiproject .org/.

SharePoint. Microsoft continues commitment to XML web services with new SharePoint products and technologies toolkits. http://www.microsoft.com/press pass/press/2004/aug04/08-09webpartspr.mspx.

Thomas, M., Mock, S., Boisseau, J., Dahan, M., Mueller, K., and Sutton, D. 2001. The GridPort toolkit architecture for building Grid portals. In *Proc. of the 10th IEEE Intl. Symp. on High Perf. Dist. Comp.*

von Laszewski, G., Foster, I., Gawor, J., and Lane, P. 2001. A Java commodity Grid kit. *Concurrency and Computation: Practice and Experience*, 13(8-9), 643-662.

WSRP1.0. Web Services for Remote Portlets specification version 1.0. http://www.oasis-open.org/committees/ download.php/3343/oasis-200304-wsrp-specification-1.0.pdf.

Wu, B., Dovey, M., Hong Ng, M., Tai, K., Murdock, S., Fangohr, H., Johnston, S., Jeffreys, P., Cox, S., Essex, J.W., and Sansom, M.S.P., 2004, A web/Grid portal implementation of BioSimGrid: a Biomolecular simulation database. *J. of Digital Information Management*, 2(2), 74-78.

Yang, X., Chohan, D., Wang, X.D., and Allan, R., 2005A. A web portal for the National Grid Service. In *UK e-Science AHM 2005, Nottingham, UK,* available on CDROM.

Yang, X., Wang, X.D., and Allan, R. 2005B. WSRP support investigation of selected open-source portal frameworks. In *GCE05: Portals Workshop, Seattle, USA, submitted to Concurrency Computat.: Pract. Exper.*