Integration of Existing Grid Tools in Sakai VRE

Xiaobo Yang, Xiao Dong Wang, Rob Allan
CCLRC e-Science Centre, Daresbury Laboratory, Warrington WA4 4AD, UK
{x.yang, x.d.wang, r.j.allan}@dl.ac.uk

Matthew Dovey
JISC Executive, 2nd Floor, Beacon House, Queens Road, Bristol BS8 1QU, UK
m.dovey@jisc.ac.uk

Mark Baker
ACET, University of Reading, Reading RG6 6AY, UK
mark.baker@computer.org

Rob Crouchley, Adrian Fish, Miguel Gonzalez, Ties van Ark
e-Social Science Centre of Excellence, Lancaster University, Lancaster LA1 4YT, UK
{r.crouchley, a.fish, m.gonzalez, t.vanark}@lancaster.ac.uk

Abstract

The integration of existing Grid tools into the Sakai VRE (Virtual Research Environment) will be discussed in this paper. In particular we describe the integration of the business logic and JSR 168 compliant portlets through presentation-oriented Web Services, via WSRP (Web Services for Remote Portlets). A set of JSR 168 compliant portlets were developed for the UK NGS (National Grid Service) Portal and have been published using WSRP4J and consumed within Sakai successfully, which proves re-use of portlets as web components is a practical option. With the help of the WSRP consumer tool, the Sakai VRE has been successfully extended to support the JSR 168 specification.

1 Introduction

The Virtual Research Environment (VRE) concept has recently been developed to help researchers manage increasingly complex tasks to help assist in today's research activities. By integrating existing resources and tools, as well as being flexible and adaptable to changing requirements, VREs can help research teams to collaborate with more versatile and flexible access to resources. A set of projects [8] has been funded by the JISC (The Joint Information Systems Committee), which aim to engaging the research community in building and deploying VREs. The Sakai VRE Portal Demonstrator is such a project, which aims to address the requirement for a single point of access to a comprehensive set of Grid and collaboration services. It is based on Sakai [10], a Collaboration and Learning Environment (CLE) for higher education developed by the University of Michigan *et al*. Sakai provides a set of collaboration tools such as chat and discussion forum. Although the JSR 168 [9] Java portlet standard is currently not supported, Sakai provides its own portal interface through which end-users benefit from single sign-on (SSO), role-based authorisation support and much more. With the help of Sakai, not only course material, but also real-time communications are now available through the Web, which helps improves the efficiency of a Virtual Learning Environment (VLE). With its innate merits, such as role-based authorisation and tool integration, Sakai was selected as the basis of our VRE project, a joint project led by the University of Lancaster, with other partners being the CCLRC e-Science Centre (Daresbury Laboratory), the universities of Oxford and Portsmouth (now Reading).

Whilst communication tools, such as a whiteboard and a audio conferencing system have been developed as native Sakai tools from scratch (Lancaster), a set of Grid tools that were previously developed for the UK National Grid Service (NGS) Portal [20] at Daresbury to provide users with seamless access to multi-site computational and data resources are now being deployed into Sakai. These tools are JSR 168 compliant portlets deployed under a customised version of StringBeans, an open-source portal framework with MyProxy logon support (see [20] for details). Al-

though, to re-design these tools for Sakai to realise the same functionality is always possible, it would be better if these NGS portlets could be remotely maintained and re-utilised directly. This is becoming even more important in the e-Science community today as there are many independent software tools and services coming out from different projects.

During the portlet development, we realised that although simple business logic can be included into portlets, a clear separation of the business logic from the portlets will benefit developers by providing a more convenient maintenance and development path. An investigation have been performed using the EJB technique [19]. Because there is no presentation layer in EJB, Sakai tools can be developed to present data provided by these EJBs. Whilst re-use of the business logic is the preferred approach in today's information systems, a further investigation showed it was possible to re-use even the presentation layer. This can be realised through presentation-oriented Web Services, e.g. Web Services for Remote Portlets [13], an OASIS specification.

In this paper, we will first discuss our recent survey on service-oriented portals and VREs. Then we provide a description of our VRE architecture and outline the two approaches for integration of business logic and the presentation layer. Here we show examples of our work. We then conclude and outline our future work.

2 Related work

As defined by Fraser [16], a *"VRE is best viewed as a framework into which tools, services and resources can be plugged."* According to this definition, the core function of a VRE is the ability to integrate existing, emerging and even future tools, services and resources. These services and resources are normally distributed and heterogeneous, which naturally suggests the idea of today's service-oriented architecture (SOA). The most important aspect of a SOA is the ability to compose new or re-factor existing software systems on top of loosely coupled services. Specifically within a VRE, besides the integration of resources, real-time communication tools are required to provide a more convenient and efficient research environment for collaboration.

In the US, *cyberinfrastructure* is used rather than *VRE*, but we consider the terms to be synonymous. The overall aim is to provide a collaborative platform with integrated services and resources that will benefit research communities. GEON [4], the Geosciences Network, is trying to build up such a service-oriented architecture for research and education in Geosciences, by linking a number of US universities, federal agencies, industry and international partners. GEON provides a set of services such as "intelligent" searching, semantic integration, and visualisation in Geoscience. Similar projects are ongoing now to develop VREs for different subjects such as CIPRES for Phylogenetics [3], IBVRE for cancer and heart modelling [7] and BVREH for humanities research [1].

Whist these projects are subject focused, VREs can be built up for general purpose. With campus grids, for example the UT Grid [11], the focus is on integrating diverse computational, visualisation, storage, data and instrument/device resources of the universities or other research institutions. The UT Grid aims to provide a comprehensive platform to support all types of research and education. Similar initiatives in the UK include CamGrid [2] and the e-Minerals mini-Grid [15].

No matter how complex a VRE could be, web-based portals are normally developed for communications between a VRE and end-users. For example, the GEON Portal is used to provide a convenient gateway to access various resources within GEON. With the help of portals, a uniform single entry-point is provided to end-users with all complex business logic and persistence layer hidden. As a web component, portlets provide a mechanism to generate markup fragments for portals to construct whole web pages. Extended from the Servlet specification, portlets brings functionality such as customisation to end-users. Advanced portals are now service oriented. The basic idea is to treat the portal as a presentation layer to *services* and *resources* inside the VRE. This brings web and Grid services together as they are treated the same way. Typical resources include databases, computational facilities, such as HPC in addition to campus Grid, and even experimental facilities. The business logic is now hidden inside the services, which normally advertise an interface using XML-based such as WSDL documents linked to a UDDI server. This makes it possible to construct loosely coupled large-scale application systems, which are platform- and language-independent.

GridPort v4.0 [6] is such a service-oriented toolkit designed to enable rapid construction of highly functional Grid portals. It simplifies the use of underlying Grid services for both developers and end-users. Similar to the NGS Portal, GridPort has a set of portlets which wrap the low-level backend Grid and information services. Through these portlets, customisable web interfaces are provided for easily making use of the underlying Grid technologies and services such as displaying resource information, scheduling jobs and transferring data. Some of the components in GridPort are designed as web services, for example, GPIR (Grid Portal Information Repository), is developed as a storage and retrieval of Grid data with web service interfaces.

Within such a service-oriented application system, the relationship among services is always data-centric. It is up to a user's client to render the data in an appropriate way. Although this brings the maximum interoperability for communications among services, it is possible to provide presentation-oriented Web Services. Such a service

returns markup fragments, for example, HTML markup, to the client. This eliminates repetitive user interface redesign work. Web portals can then construct web pages using mark-up from either local or remote portlets.

3 Sakai Demonstrator VRE architecture

The Sakai VRE Demonstrator project is aiming to building up a general-purpose demonstrator using the Sakai framework. To meet the requirement of sharing tools and services among different projects, the Sakai VRE is defined as a pluggable architecture, which makes it possible to "plug" existing, emerging and even future tools and services into the framework.

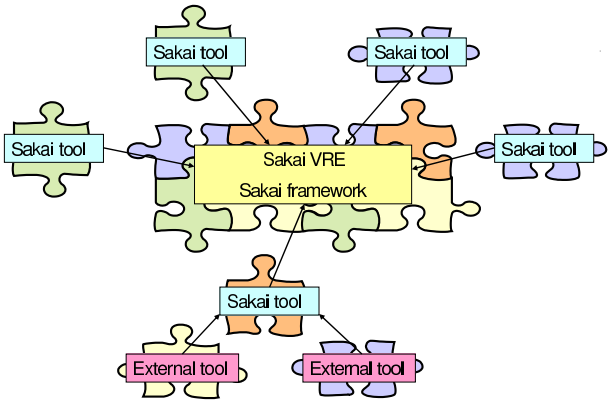

Figure 1. Pluggable Sakai VRE architecture.

In Fig. 1, tools are shown directly plugged into the Sakai framework to extend its capability. More importantly, a tool can be developed to enable integration of external tools that Sakai does not understand. Such an agile architecture makes it possible to support standards additional to those native to Sakai and also to adapt to future standards.

In the next section we give two examples: 1) extending Sakai by developing a native tool; 2) extending Sakai to support the JSR 168 portlet standard through WSRP.

4 Integration of Grid tools in Sakai

Portlets that can support tasks such as proxy credential management, remote job submission and monitoring have been developed in the Grid Technology Group at Daresbury Laboratory. These portlets have been successfully deployed in the UK National Grid Service Portal and cloned in other portals, for example the IBVRE. In some sense, a VRE is in fact built on top of Grid, which is a good candidate for integration of distributed resources. Whilst not directly integrated with Sakai, there is still a need to access Grid tools similar to those developed for the NGS Portal. In this section, we will discuss two approaches to make use of our existing work.

4.1 Integration of business logic

A practical approach in software development is to separate the business logic from the presentation. This also applies to portlet development. The first approach we are going to discuss is to re-use the business logic developed for portlets. In this example, the business logic is implemented as some Enterprise Java Beans (EJBs), see Fig. 2. These EJBs can then be utilised to construct either portlets or Sakai tools. In EJB 2.1, stateless session beans can also be exposed as Web Services. This brings more flexibility for other platforms to link to J2EE components, which is important for construction of a SOA.

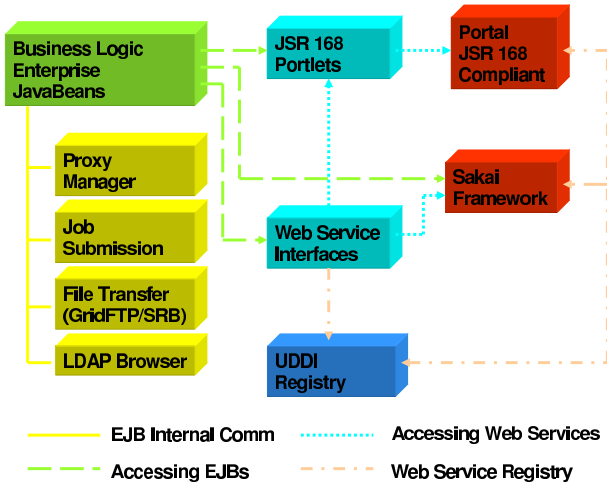

Figure 2. Integration of business logic.

A LDAP browser (see Fig. 3) for querying the NGS Grid information service has been developed as a native Sakai tool, which makes use of a LDAPQueryBean deployed inside JBoss, the chosen J2EE application server. A detailed discussion of applying the J2EE/EJB technologies in portal/portlet development for Grid users was described in [19].

Although, in this example EJBs are used, the business logic represented in other formats, for example Web Services and as Java Objects (POJOs) can follow the same procedure as all of them are data-centric. Web Services provide a good opportunity to keep software components language- and platform-independent. This is vital for constructing large-scale complex service-based systems. For example, after several years of development, many e-Science projects now have their own special software components to solve particular scientific problems in area such as Biology or

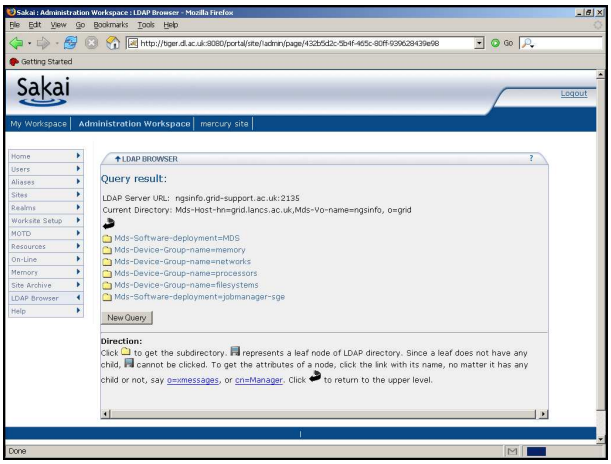

Figure 3. LDAP browser tool running in Sakai.

Chemistry. If these components are designed as, or converted to Web Services, it would be easy to construct more complex software systems, potentially supporting multi-disciplinary collaborations.

4.2 Integration of both business logic and user interface

4.2.1 JSR 168 portlets

Whilst exposing software components as Web Services makes it possible for different software systems to communicate, user interfaces are normally required for those services to interact with end-users. A portal with portlets provides a customisable user interface for presenting information. The portlet specification, JSR 168 [9] was designed to solve the interoperability issue between portlets and portlet containers. This has been proven to work fairly well as we have moved portlets developed using StringBeans to other portal frameworks such as GridSphere.

To solve the interoperability issue between portal frameworks, another portal standard, Web Services for Remote Portlets (WSRP), was released by OASIS in 2003. As mentioned earlier, Sakai does not support JSR 168, therefore it cannot consume JSR 168 portlets directly, rather only EJBs or Web Services exposed as Sakai tools. Unfortunately, this means that existing portlets could not easily be used in Sakai. For this reason we took the decision to write a WSRP consumer for Sakai. When equipped with this consumer it is possible for Sakai to consume portlets published by WSRP producers, which are Web Services. Furthermore, Sakai has the ability to integrate existing web programs, such as servlets and JSF (JavaServer Faces) applications. It would even be practical to develop a WSRP consumer outside of Sakai. Then it can be integrated within Sakai with only few modifications to meet the Sakai's requirements.

4.2.2 Web Services for Remote Portlets

WSRP makes use of the web service concept to transfer markups from a remote portlet container. Therefore it benefits from aspects of traditional web services such as platform- and language-independence. *Producer* and *Consumer* are two important actors defined in the WSRP 1.0 specification [13]. While taking the responsibility for managing portlets, a producer provides a set of web service interfaces through which a consumer can then access its portlets. There are altogether four interfaces defined, with two of them optional.

Self Description: a required interface provides metadata for a consumer to interact with each portlet the producer hosts. It also gives information about the producer's capabilities.

Markup: a required interface for interacting with user requests and generating markup fragments.

Registration: an optional interface to set up a relationship between a producer and a consumer. A producer could response to a consumer according to its capabilities.

Portlet Management: an optional interface used for both managing the life-cycle of hosted portlets and portlets' persistent state.

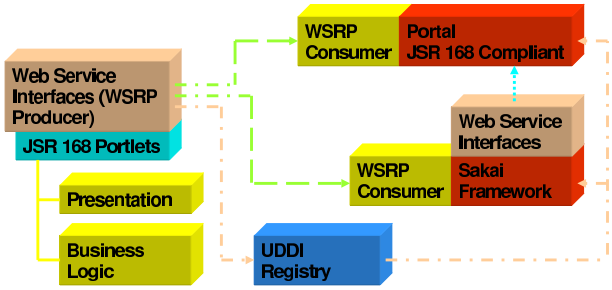

Figure 4. Integration of both business logic and user interface.

As illustrated in Fig. 4, a WSRP producer is modelled as a portlet container, while at the same time it publishes its Web Service interfaces. A consumer sitting between end-users and a producer acts as a broker. Such a consumer is responsible for collecting end-user's requests and re-directing them to the corresponding producer. Once a response is received, the consumer has to handle and render it for end-users. WSRP consumers can be attached to

other frameworks, for example portals and Sakai, as shown in Fig. 4. Once these frameworks have the ability to handle the markup provided by consumers, they are able to consume remote portlets as easily as local ones.

4.2.3 WSRP sequence flow

As mentioned early, the WSRP 1.0 specification defines *Producer* and *Consumer* to complete the remote portlet's publication and consumption. In this section, a description of the sequence flow is given for better understanding of how producer and consumer work together to complete the portlet's exposition and consumption.

Fig. 5 gives a simplified sequence flow of WSRP communications. In this figure, communications such as *clonePortlet* and *deregister* are omitted to make it more readable. A detailed description of WSRP communications between consumer and producer can be seen in the WSRP 1.0 Primer [12] and other introductory materials [17]. A consumer first talks to the producer's description interface to retrieve metadata. This includes a description of the producer and portlets it handles. During this stage, the consumer may be asked to register itself. According to the consumer's capabilities, the producer may provide markup in different formats. The consumer can then select the portlet it wants to access. A unique portlet handle returned for communications between producer and consumer so that the portlet can be identified. The producer will first provide a default view of the portlet. The consumer needs to add to the markup fragment so that a full (usually HTML) web page can be constructed and rendered. The consumer also needs to collect requests, such as form input, and redirect them to the producer. This is the most important task of the consumer. Only when a user request is transferred correctly, will a correct response be produced. As reported in [21], many WSRP consumers could not handle such a redirection correctly which makes them currently unpractical.

4.2.4 WSRP consumer for Sakai

Although WSRP support is claimed by many portal frameworks, it is currently far from being mature in open-source portal frameworks, such as the eXo platform and Liferay, due to the complexity of the specification [21]. Interoperability between producers and consumers, especially from different vendors, is still immature.

Based on the sequence flow shown in Fig. 5, a servlet-based web application has been developed on top of a ProxyPortlet, an Apache WSRP4J [14] reference consumer. Currently portlet metadata, such as name and handler, is published using the service description interface. The consumer needs to know each producer's interface to get the metadata. As end-users are normally not interested in these

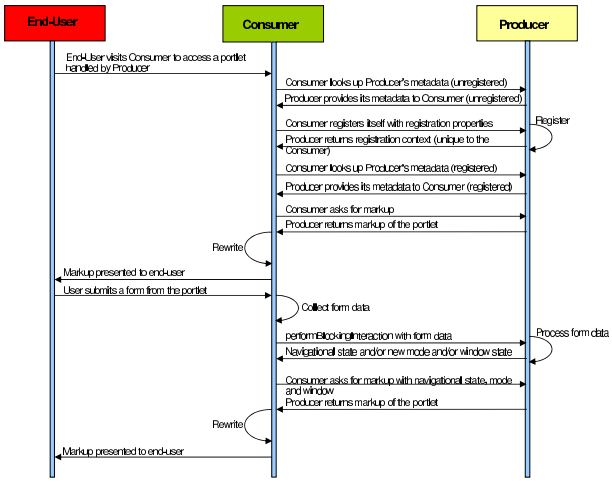

Figure 5. Simplified WSRP sequence flow.

Web Services interfaces, a more flexible approach is to publish the metadata to central UDDI registry where producers together with their portlets can be registered [18]. A servlet has been written to execute such a query, and to get a list of remote portlets and related producer interface URLs that may meet the end-users' requirements.

This WSRP consumer has been ported to Sakai and acts as an embedded tool. It has been successfully tested to access the NGS Portal portlets published by a WSRP4J producer. In order to execute Grid tasks, for example to submit a job to a remote computing resource, a user first retrieves his proxy credential using the ProxyManager portlet from the UK Grid Support Centre MyProxy server. The credential is then stored in the session, which is shared by different portlets within the same portlet application. Once a job is finished, the user can transfer his data to another server using the FileTransfer portlet, see Fig. 6.

Referring to Fig. 1 - assume that the "Sakai tool" below the "Sakai framework" is a WSRP consumer, then those external tools are remote portlets. Although not necessarily JSR 168 portlets, JSR 168 is supported if the WSRP producer makes use of a JSR 168 portlet container. For instance WSRP4J, used in our example, makes use of Pluto's reference implementation of JSR 168. In this way the Sakai framework has been extended to support JSR 168 through WSRP.

As mentioned earlier, e-Science has already produced a lot of software components in different domains. The service-oriented architecture is becoming dominant for the construction of large-scale systems, such as VLEs and VREs. It is, therefore, recommended to adopt Web Services at the minimum if possible, and may be move to Grid services, such as WSRF (Web Services Resource Framework) in the future. Furthermore, if these components are exposed

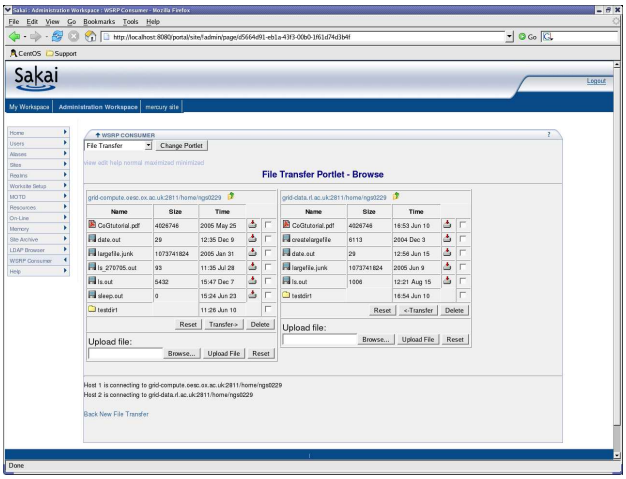

Figure 6. GridFTP-based file transfer portlet running in Sakai through WSRP.

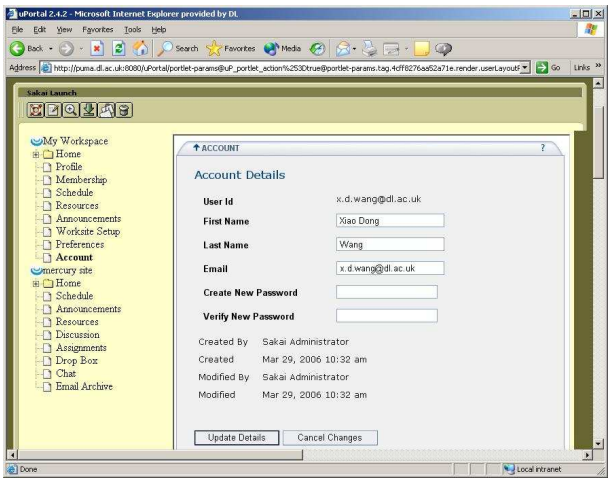

Figure 7. Sakai JSR 168 portlet running inside uPortal.

as JSR 168 portlets, developers will also benefit by not having to re-designing the complex UIs, but use WSRP as it matures – plug-and-play complex web portals will then become real. Besides all these benefits, WSRP is in fact not limited to re-usage of JSR 168 portlets, since it makes use of normal Web Services, which are language independent. An investigation into developing a Perl-based WSRP implementation has been performed by the Go-Geo! Project team [5]. This opens up a promising approach for integrating existing applications.

4.2.5 Integration of Sakai in JSR 168 portal frameworks

This article has so far talked about the integration of JSR 168 portlets into the Sakai framework, this is however only one side of the scenario. On the other side, there is a need to integrate Sakai tools in JSR 168 portal frameworks which may already be in use by e-Science projects. Sakai has focused on collaboration and learning and it glues together a number of collaborative tools which can be very useful in constructing fully-functional web portals. The Sakai development team has released a JSR 168 portlet by which the Sakai data structure can be used through web service interfaces and their output rendered in portal frameworks like uPortal and GridSphere. This gives an approach to integrating Sakai collaboration tools and even workspace with portals. Fig. 7 shows a user is editing his Sakai account information within uPortal through the portlet mentioned above.

Sakai 2.1 also provides a WSRP producer to expose some of its tools. This gives yet another approach for integration of these tools in portal frameworks which have working WSRP consumers. Using our Sakai WSRP consumer it is also possible to expose tools hosted in one Sakai instance in a separate Sakai instance.

5 Conclusions and future work

In this paper, re-use of existing Grid tools in the Sakai VRE Demonstrator Project has been discussed. Besides the traditional re-use of business logic, a presentation layer has also been successfully integrated into Sakai. This has been done using WSRP to consume remote JSR 168 compliant portlets that were originally developed for the UK NGS Portal. This approach eliminates the effort of re-designing user interfaces for different projects, and makes re-use of the underlying core services.

Our prototype work on integrating existing software components through WSRP highlights a promising approach for today's e-Science projects. If components can be exposed as Web Services or JSR 168 portlets, they can in the future be re-utilised without much effort. With the help of the WSRP consumer tool, Sakai has effectively been extended to support the JSR 168 specification.

Whilst portal frameworks are adopting WSRP, it is still immature especially when considering producers and consumers are from different vendors. WSRP also does not solve all the issues; for example, in the WSRP 1.0 specification, security is not touched upon, except by making use of SSL/TLS or Web Services security mechanisms. Inter-portlet communication is also not addressed. Also there is issue with file upload/download function in a portlet, for example the FileTransfer portlet shown in Fig. 6, since remote

portlets are no longer facing end-users directly. Some of these issues are now under consideration in WSRP 2.0.

Our future work will include securing remote portlets using mechanisms such as Shibboleth, although this will probably require Shibboleth 2.0 to address the N-tier authentication/authorisation issues. More advanced registration interfaces may also be considered to provide customised user interfaces for different clients. Further investigation is also required to determine how mature it is to integrate Sakai in JSR 168 portal frameworks.

Acknowledgements

We thank the anonymous reviewers for their insightful comments helped to improve this paper. This work was undertaken at the CCLRC e-Science Centre, Daresbury Laboratory supported by UK JISC (The Joint Information System Committee).

References

[1] BVREH: Building a Virtual Research Environment for the humanities. *http://bvreh.humanities.ox.ac.uk/*.

[2] CamGrid: Building a university-wide grid across the University of Cambridge. *http://www.escience.cam.ac.uk/projects/camgrid/index.html*.

[3] CIPRES: Cyberinfrastructure for phylogenetic research. *http://www.phylo.org/*.

[4] GEON: The geosciences network. *http://www.geongrid.org/*.

[5] Go-Geo! portlet work. *http://www.gogeo.ac.uk/geoPortal10/PortletInfo.html*.

[6] GridPort. *http://gridport.net/*.

[7] Integrative Biology Project. *http://www.integrativebiology.ox.ac.uk/*.

[8] JISC Virutal Research Environments Programme. *http://www.jisc.ac.uk/index.cfm?name=programme_vre*.

[9] JSR-168 Portlet Specification. *http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html*.

[10] Sakai: Collaboration and learning environment for education. *http://sakaiproject.org/*.

[11] UT Grid Project. *http://utgrid.utexas.edu/*.

[12] Web Services for Remote Portlets 1.0 Primer. *http://www.oasis-open.org/committees/download.php/10539/wsrp-primer-1.0.html*.

[13] WSRP Specification 1.0 by OASIS. *http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specication-1.0.pdf*.

[14] WSRP4J. *http://ws.apache.org/wsrp4j/*.

[15] M. Dove, E. Artacho, T. White, R. Bruin, M. Tucker, P. Murray-Rust, R. Allan, K. K. van Dam, W. Smith, R. Tyer, I. Todorov, W. Emmerich, C. Chapman, S. Parker, A. Marmier, V. Alexandrov, G. Lewis, S. Hasan, A. Thandavan, K. Wright, C. Catlow, M. Blanchard, N. de Leeuw, Z. Du, G. Price, J. Brodholt, and M. Alfredsson. The eMinerals project: Developing the concept of the virtual organisation to support collaborative work on molecular-scale environmental simulations. In *UK e-Science AHM 2005, available on CDROM*, Nottingham, UK, September 2005.

[16] M. Fraser. Virtual Research Environments: Overview and activity. *Ariadne Magazine*, 44, July 2005.

[17] P. Gupta. WSRP: Dynamic and real-time integration. *Web Services Journal*, 5(8):10–19, August 2005.

[18] X. D. Wang, X. Yang, and R. Allan. Plug-and-play remote portlet publishing. In *GCE05: Portals Workshop*, Seattle, USA, November 2005.

[19] X. Yang, A. Akram, and R. Allan. Developing portal/portlets using Enterprise JavaBeans for Grid users. In *GCE05: Portals Workshop*, Seattle, USA, November 2005.

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

[21] X. Yang, X. D. Wang, and R. Allan. Investigation of WSRP support in selected open-source portal frameworks. In *GCE05: Portals Workshop*, Seattle, USA, November 2005.