

Virtual Research in the UK: Advanced Portal Services

Mark Baker and Hong Ong

Distributed Systems Group, University of Portsmouth, UK

Rob Allan and Xiao Dong Wang

e-Science Centre, CCLRC Daresbury Laboratory

Abstract

As part of e-Science work being undertaken on several projects at the University of Portsmouth and Daresbury Laboratory, we are investigating and developing advanced Grid portals using a range of emerging portlet-based technologies and services. The OGSA Testbed project, led by the University of Portsmouth, has enabled us to evaluate the use technologies such as GT3, OGSA-DAI, and eventually WSRF in building a variety of portlets to expose remote services to users with a variety of application needs.

1. Introduction

In this paper we describe the portal technologies and services that are being developed, extended and implemented to support e-Science groups with whom we are working. In the first part of the paper we describe our application areas as well as our original motivation for taking up portal technologies; this part of the paper also includes a critique of some of the hurdles that needed to be overcome. We then move on to describe a range of emerging portlet toolkits and services and outline how these can be used to fulfil the needs of our particular applications. In the second part of the paper we detail our experiences using these emerging technologies; here we also provide examples and discuss implemented services. In the final part of the paper we summarise the paper, and draw a number of conclusions about using portlets and Grid services to deliver an e-Research environment and outline our future work plans.

1.1 Early Science Portals

A typical first generation portal is shown in Figure 1. They shared following characteristics [1]:

- A three-tiered architecture, consisting of an interface tier of a Web browser, a middle tier of Web servers, and a third tier of backend services and resources, such as databases, high performance computers, storage, and specialised devices.
- A user makes a secure connection from their browser to a Web server.
- The Web server then obtains a proxy credential from a proxy credential server and uses that to authenticate the user.

- When the user completes defining the parameters of the task they want to execute, the portal Web server launches an application manager, which is a process that controls and monitors the actual execution of task(s).
- The Web server delegates the user's proxy credential to the application manager, so that it may act on the user's behalf.

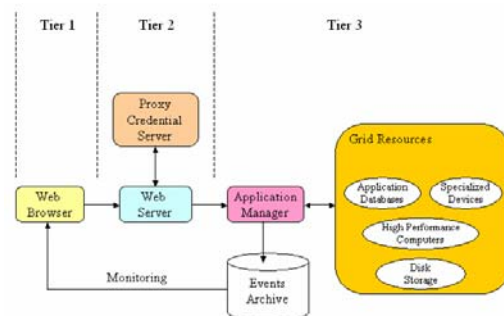


Figure 1: Grid Portal Architecture

In some systems, the application manager publishes an event/message stream to a persistent event channel-archive, which describes the state of an application's execution and can be monitored by the user through their browser.

First generation portals were limited and suffered from the following:

- **Lack of Customization** - developers instead of users normally built portals because the knowledge and expertise needed a portal toolkit was beyond the capability of most end users. End user access to the Grid via a portal was statically configured and it was almost impossible to dynamically customize a portal to meet

their specific needs, e.g., to add or remove some portal services.

- **Restricted Grid Services** – early portals were tightly coupled with specific Grid middleware technologies such as Globus, which results in restricted services. It was hard to integrate services provided by different technology providers.
- **Static Grid Services** – whereas a Grid environment is dynamic in nature with many evolving service. Early portals could only provide static services and lacked facilities to easily expose newly created services to users.

While there are limitations with first generation portals and toolkits, the experiences and lessons learned developing portals have paved the way for the development of more sophisticated and user-friendly portals.

To overcome the limitations of earlier portals, portlets have been introduced and promoted for use in building portals. From a user's perspective, a portlet is a window (see Figure 2) in a portal that provides a specific service, for example, a calendar or news feed. From an application development perspective, a portlet is a software component written in Java, managed by a portlet container, which handles user requests and generates dynamic content. Portlets, as pluggable user interface components, can pass information to a presentation layer of a portal system. The content generated by a portlet is also called a fragment. A fragment is a chunk of markup language (e.g., HTML or XHTML) adhering to certain rules and can be aggregated with other fragments to form a complete document. The content of a portlet is normally aggregated with the content of other portlets to form the portal page. A portlet container manages the lifecycle of portlets in a portal.

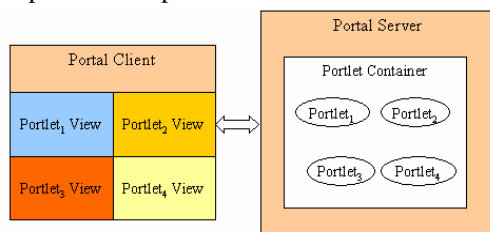


Figure 2: A Portal with four portlets

A portlet container provides a runtime environment in which portlets are instantiated, executed, and finally destroyed. Portlets rely on the overall portal infrastructure to access user

profile information, participate in a presentation window, and communicate with other portlets to access remote content, lookup credentials, and store persistent data. A portlet container manages and provides persistent storage mechanisms for portlets.

A portlet container is not a stand-alone container like a Java Servlet container; instead, it is implemented as a layer on top of the Java Servlet container and reuses the functionality provided by the Servlet container.

Currently, there are two groups that are working on the standardization of portlets. One is OASIS (Organization for the Advancement of Structured Information Standards) [2] with their Web Services for Remote Portlets (WSRP) specification [3]. The other is JCP (Java Community Process) [4] and their JSR 168 specification [5]. The portlet JSR-168 (ratified in August 2003) specification handles the presentation end of information enabling re-use of portlets in different containers. In order for containers to present their contents as services IBM is taking the lead on WSRP, the Web Services for Remote Portlets standard (also ratified in August 2003), which is based on XML and Web services. WSRP will allow portals to retrieve content from other portals via their portlet containers and other data sources.

The advantages of a portlet-based architecture are that, principally each underlying function or service can be associated with a unique portlet. This makes it easy to add new services, and many different groups can then independently contribute portlets, which can be plugged into the portal. Using WSRP they can be distributed and managed remotely on many servers and the portals described by WSDL-like information. Each user can select and configure the portlets they wish to use and the selection can become part of a persistent “context”. For instance certain portlets may only be useful for expert or administrative users and can be discarded by others or have their access controlled via a role-based mechanism.

2. Applications and Motivation

2.2 LCGPortal

The Distributed Systems Group at the University of Portsmouth, as part of its OGSA Testbed project is migrating its Liquid Crystal Portal [6][7], which is a stovepipe solution that used the Globus Toolkit version 2 (GT2),

MyProxy, GridPort tools, and a customized JavaScript-based Web interface. The portal provides all the services that a developer would need to configure, manipulate, submit, and view liquid crystal simulations. The updated portal uses range of emerging portal utilities and tools, including GridSphere [8], in-house portlets, and portlets from the Alliance Science Portal [9] which now form the basis for the OGCE portal supported by the US NSF Middleware Initiative [10] including the Proxy Manager to implement GSI, GridFTP for various communications and the GRAM Job launcher to execute liquid crystal simulations and render output results.

GridSphere is based on IBM's WebSphere and provides a "white-box" framework in which users can override base classes and "hook" in their own methods. It therefore requires users to become familiar with core framework interfaces, which are however based on the standard JSR-168 API. Extensive use is made of design patterns, which provide template solutions to commonly recurring software design problems. GridSphere also provide a common language that makes code easier to read and understand. The Model View Control (MVC) paradigm is used to separate logic from presentation as in other portlet frameworks.

Features of GridSphere include:

- Portlet API implementation,
- Support for the development and integration of "third-party portlets",
- A higher-level model for building complex portlets using visual beans and the GridSphere User Interface tag library,
- An XML based portal presentation description that can be modified to create customized portal layouts,
- Built-in support for Role Based Access Control separating users into guests, users, administrators and super users,
- A portlet service model that allows for the creation of "user services", where service methods can be limited according to user rights,
- Persistence of data provided using Castor JDO from ExoLab for RDMS database support, SQL and OQL.
- Integrated Junit and Cactus unit tests for server side testing of portlet services including the generation of test reports,
- Documentation uses DocBook for HTML and PDF output of guides and tutorials

- GridSphere core portlets which offer base functionality including login, logout, user and access control management,
- Localisation support in the Portlet API implementation and GridSphere core portlets support English, German, Czech, Polish, Hungarian, and Greek.
- Open-source and 100% free.

2.3 HPCPortal and InfoPortal

At Daresbury similar portlet technology is being used and embedded into the Apache JetSpeed framework [11] to implement Grid Information Services. These are based on the type of service prototyped in InfoPortal [12], a generic free-to-use Web-based Grid information system developed at CCLRC Daresbury Laboratory for the UK e-Science Grid. It provides a variety of presentations and search facilities, which can include direct links to information accessible from remote resources. InfoPortal v1.0 used standard Perl/CGI Web technology and was closely modelled on HotPage, but used the Globus Monitoring and Discovery Service plus an XML database as sources of information about Grid resources. InfoPortal v2.0 uses PHP-Nuke as a content management framework and contains the collaborative tools that come with it. The new services are being developed using Web and Grid service toolkits, including Globus GT3, OGSA-DAI, and UDDI [13]. In the longer term, attention will focus more on the new WSRF standard and on porting more Grid services, which are currently expressed through the HPCPortal [14] for application end users. Examples of InfoPortal are shown in Figure 4 and Figure 5.



Figure 3: The DSG GridSphere Portal

For this work it is likely that the CHEF framework [15] will be used since it already includes a number of tools, which have been developed in an educational environment.

CHEF is a collaboration tool aimed equally at course administrators and students. It offers administrators the facility to set up a course work-site, and students to set up study work-sites. Each work-site can be configured with a number of included tools like chat, a discussion board, shared calendar, and communal file space. It makes use of the Jakarta Jetspeed portal framework to present these tools to the user in a collection of portlets.

They are equally applicable to sharing resources for research. CHEF already forms the basis for the OGCE portal. This development work will underpin projects such as e-HTPX [16], e-Minerals, e-Materials [17] and ReDRESS [18] which are being presented in other papers at the conference.

ReDReSS, Resource Discovery for Researchers in e-Social Science, is a JISC-funded project for awareness and training in computational and e-Science methods. This started in November 2003 and is currently integrating content into a portlet-based framework. CHEF is being used for the prototype.

3. Portal Services

Within a portal a number of internal services are needed to address issues of the coordination of tools (portletlets) within an overall framework. Methods can be provided as an "internal" class library, which resides alongside the portlet and service APIs (the model part of the MVC paradigm). Each portal framework could have the same, or a different set of tools, but the way they are integrated may differ between user groups. Alternatively the services could be federated and available via Web Services calls to specialised servers elsewhere in a virtual research environment.

These portal services mostly imply research issues. Some simple ones, such as managing the look and feel of the portal, personalisation, and accessibility are provided directly by the portlet container. Some example portal integration services are now listed:

- **Session Management** involves the management of a session key and related issues. It requires database access for storing and retrieving other items relevant to the session. User can authenticate and start a new session or revert to a previous one. The service can open and close sessions and log the state of a session from.

Features like rollback and replay, including personal workflow, can also be available.

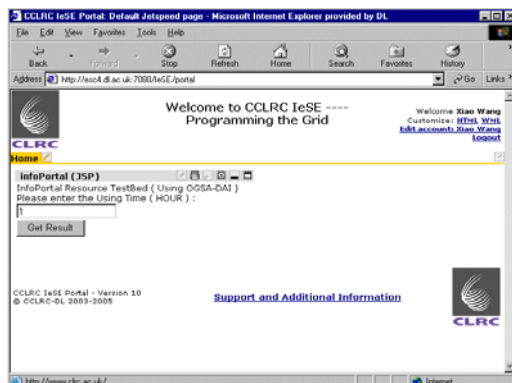


Figure 4: Portal Interface (a)

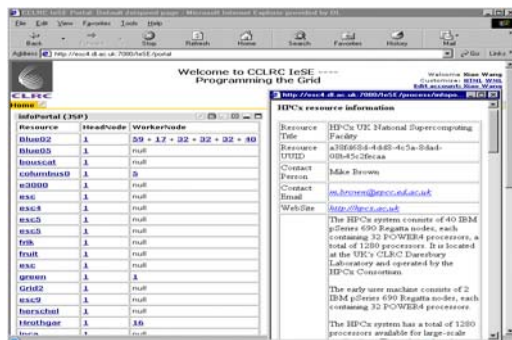


Figure 5: Portal Interface (b)

- **Authentication** using MyProxy, which is a repository of valid proxy certificates for authenticated users. The portal can download these for delegation to trusted external services. A service can also check that certificates are, for example, still valid and refresh them if not. Part of the integration API would allow storing and retrieval of the proxy into the portal database for later use. This will be done using a session key and UID (e.g. DN or unique e-mail address). Having the certificate associated with the session key allows authorisation issues to be tackled, e.g. using subsidiary certificate or another method.
- **VO Management** that could for example be based around a project (as described by UDDI), which would typically have its own portal and Grid. VO users will have been authenticated and have received a digital identity (certificate). They are then given rights based on the roles they are taking in this VO and thus can be authorised to access services.

- **Integrated State** is related to the need to manage data related to state information for a portlet UID. There is a general need to develop the concepts related to integrated state. For example:
 - State can be used as an event trigger,
 - State needs to be logged for session management or workflow,
 - What states can portlets and services have which are meaningful for rollback and replay?
 - Service and Portlet location, which can be published, queried, and looked up in a registry. This also requires semantic support as it is import to annotate service information with further information such as what the service does and why.
- **Portal Preferences**, which can be built up from a "preferred set" of services and portlets and be based on usage. This service can also log semantic information and build a related ontology. The service extends the idea of a workspace toolset allowing dynamic semantic/function-driven choice.
- **Semantic/Ontology Support** for information about services and portlets in the framework. These services will be used for decision support and choice, augmenting stored preferences. These services would not cover generic semantic issues, which would need separate tools.
- **Workflow** via directed links between components (typically graph based). An event mechanism is used to trigger actions within portals and attached services. The graphs within the portal will be mostly pre-defined, but with constrained facilities to swap in and out components and provide additional inputs at decision points.
- **Trails and Personalisation** could involve logging of usage for off-line mining and analysis, e.g. for developers to improve presentation, ease of use, and optimisation.
- **Inter Portlet Communication and Event Management** will provide message-based communication mechanism between portlets, possibly with event triggers and asynchronous handlers.

Some key research issues in implementing the aforementioned services include:

- Identification of user/session/portlet/services, these are typically name value pairs for the

sessionKey, UID, portletID, and serviceID. Similar to the information saved in a cookie in a 1st generation portal. In addition non-portal tools could also use these by creating method calls to use the information.

- State definitions are pre-defined set of states that need to be identified. Such identification could be the key to using the event mechanisms and session logging.

4. Experiences

Based on an analysis of the aforementioned issues, we have attempted to implement a number of portal services using a distributed architecture based on Web and Grid services' interfaces and the appropriate middleware. Many of the portal services that have been identified, in addition to those such as the Grid Information Service, require access to local, remote, or federated databases. In this situation we have used OGSA-DAI to link to an XML or relational database.

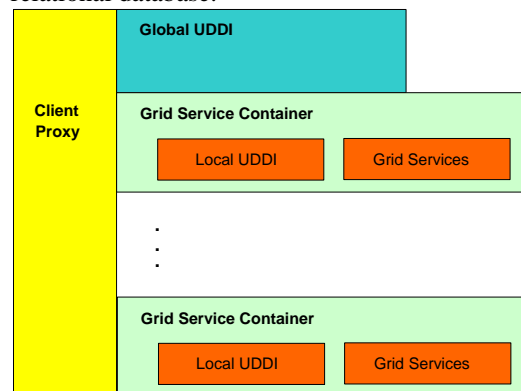


Figure 6: Components of Service Architecture

We first considered the case of augmenting an interface to a UDDI registry with a descriptive XML naming schema used for the contact information, see [19]. For processing and coordinating resource information and computational tasks, we deployed a Grid-based system for information and data access.

In this system a Global UDDI registry records for the local UDDI handles of the Grid service containers, which are hosted in public nodes. Therefore the Grid services, which are registered in the local UDDI, can be found and called by a remote client (see Figure 6).

As a first template, we developed some local portlets for Globus-based services, which could present resource and service state information. The portlets were written using the MVC (Model-View-Control) structure (see Figure 7).

The Model here will invoke a Grid service, which uses handles obtained from the UDDI registry

Secondly we considered a portlet presenting Grid service state information to users. The Grid resource information portlet offers access to local, remote, or federated databases. OGSA-DAI is used along with a person XML naming schema [19] used for contact information. The link between the UDDI and local database is mediated by the portlet.

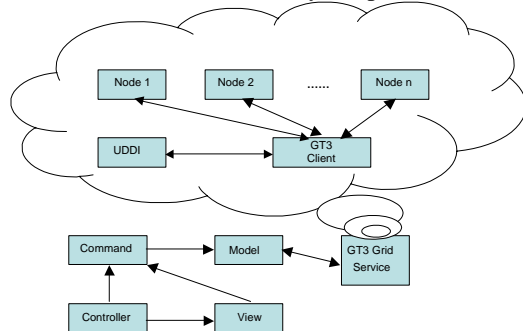


Figure 7: Portlet MVC Architecture with Grid Services

A state information portlet offers the information about the status of machine referenced by their public nodes. The GT3 *MasterForkManagedJobFactoryService* handle is used for the source of this information and the sink is the GT3 *IndexService* handle. So by invoking *IndexService*, users can get a node's state information. See Figure 8 for an example of this service.

5. Conclusions

A portal provides a unified web-based gateway to a range of grid and other services. First generation portals were tightly coupled with Grid middleware technologies and could only provide static and restricted services. These portals lacked the ability to be customised by end users to meet their specific needs. The Grid community is increasingly taking up Portlet technologies to meet their needs.

The current generation of Grid portal is focused on portlets that support user customisation; so that users can put together their personalised portals. Portals of this generation can provide extensible and dynamic Grid services.

The current generation of Grid portals are developed using pluggable components known as portlets; which execute inside a portlet container. Portlets can be added, or removed

from a portal, thus providing users with the ability to customize the services they require at the portal level. A Grid portal built from portlets can provide users with the ability to integrate services provided by different middleware technologies.



Figure 8: Portlet showing Grid Resource Information via Service Data Index Service

Portlets have the following benefits compared with earlier portals.

- **Portal Customisation** – A user, instead of a system developer, can construct their personalised portals out of the available portlets to meet their specific needs. Portlets can be dynamically added or removed from a portal at runtime.
- **Extensible Grid Services** – Portals built from portlets are loosely coupled with middleware technologies since services can be exposed as standard portlets. A portal constructed from portlets provides users with the ability to integrate services from different service providers.
- **Dynamic Services** – new services and components are being developed for the Grid. A portal should be able to provide users with the ability to access these dynamic services. To this end, a mechanism can be provided to expose services as individual portlets that can be published and accessed via a portal.

It is clear that there is a lot of established expertise and momentum in the UK to develop Web-based portals for a variety of purposes. We have established strong links and potential collaborations bridging the UK, USA, and other European developers as well as also bridging the e-Science and JISC communities. It is important to continue this work and lead the identified areas, which will be taken via the Global Grid Forum research, and working groups as input into the definition of standards leading to software sharing.

References

- [1] Dennis Gannon, et al, Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications. Cluster Computing, 5(3): 325-336 (2002), <http://www.extreme.indiana.edu/~gannon/ProgGrid/ProgGridsHTML.htm>
- [2] OASIS, <http://www.oasis-open.org>
- [3] WSRP, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
- [4] JCP, <http://www.jcp.org>
- [5] JSR 168, <http://www.jcp.org/jsr/detail/168.jsp>
- [6] M.A. Baker, H. Ong and G. Smith, A Status Report on developing a Grid-enabled Web Portal using Commodity Toolkits, University of Portsmouth, 2001, <http://esc.dl.ac.uk/GridWG>
- [7] M.A. Baker, H. Ong and G. Smith, A Portal for Liquid Crystal Modelling Applications, University of Portsmouth, 2002, <http://esc.dl.ac.uk/GridWG>
- [8] GridSphere, <http://www.gridsphere.org/>
- [9] Alliance Science Portal, <http://www.extreme.indiana.edu/xportlets/>
- [10] OGCE NMI-portal, <http://www.ogce.org/>
- [11] Jetspeed, <http://jakarta.apache.org/jetspeed>
- [12] InfoPortal, <http://www.grid.ac.uk/InfoPortal>
- [13] R.J. Allan, D. Chohan, X.D. Wang and M. McKeown, UDDI and WSIL for e-Science Grid Support Centre, 2002, <http://esc.dl.ac.uk/WebServices>
- [14] HPCPortal, <http://esc.dl.ac.uk/HPCPortal>
- [15] CHEF, <http://www.chefproject.org>
- [16] e-HPTX, <http://www.e-htpx.ac.uk/>
- [17] e-Science Centre, <http://www.e-science.clrc.ac.uk>
- [18] ReDReSS Portal Services for Awareness and Training, Rob Crouchley, Adrian Fish, Rob Allan and Dharmesh Chohan, Procs. AHM 2004, Nottingham.
- [19] UDDI Person Naming Schema, <http://thames.dl.ac.uk:8080/index-service/person.xsd>