

Grid Middleware Portal Infrastructure

Dharmesh Chohan
Senior Software Developer
CCLRC e-Science Centre
Daresbury Laboratory
+44 1925 603790
d.b.chohan@dl.ac.uk

Asif Akram
Software Developer
CCLRC e-Science Centre
Daresbury Laboratory
a.akram@dl.ac.uk

Rob Allan
Group Manager
CCLRC e-Science Centre
Daresbury Laboratory
+44 1925 603207
r.j.allan@dl.ac.uk

ABSTRACT

This paper gives a description of building a Grid middleware portal infrastructure in CCLRC to allow computational scientists, researchers and application users access to resources via an easy to use Web based portal interfaces. The goal is to develop common Grid application components that can be used by portal developers and administrators to build and deploy on open source portal frameworks. This will allow users to authenticate securely to remote resources and provide transparent access via Grid related tools to manage their tasks efficiently. The current portal technology, JSR 168, is described for the development of Java based applications. The work is substantiated by giving an example of building the UK National Grid Service (NGS) portal. NGS Portal itself works as middleware; it “glues” different Grid Resources together in a single location. Portals and portlets are not confined to a Web-based solution, through Web Services for Remote Portlets WSRP [8] portlets offered by different portal frameworks can be consumed in software ranging from desktop to PDA.

Categories and Subject Descriptors

D.3.3 [Programming Languages]: Language Constructs and Features – *Concurrent programming, Distributed programming.*

General Terms

Algorithms, Management, Design, Reliability, Security, and Standardization.

Keywords

Middleware, Portals, Portlets, Grid, Web Services.

1. INTRODUCTION

There are many definitions of what constitutes a Grid. Here we define briefly a Grid as a collection of heterogeneous distributed interconnected computing and data resources that provides a large scale virtual resource with an appropriate single user interface. The Globus Toolkit [2] is middleware which is widely

used to build such Grids and Grid-enabled applications today. Most of our Grid work uses the Globus Toolkit, and is mainly to support the academic research community. CCLRC is involved in a number of Grid deployment projects such as the UK e-Science Grid [6], North West Grid [22] and the National Grid Service [1].

Portals and Portlets are emerging technologies currently gaining a lot of popularity. Open source portal frameworks are useful for programmers due to their ease in development, richness in functionality, customization of interface and a natural interface to a pluggable architecture such as the Services Oriented Architecture (SOA). The portal framework and portlet technology is further discussed below.

Within CCLRC, the NGS portal is now being used as a template to “clone” portal environments for other projects to benefit from taking advantage of the Java portlet technology.

2. Portals and Portlets

Portals and Portlets are emerging technologies with improving specifications and enhanced support from both open source and commercial software companies. A portal is a Web-based application that acts as a gateway between users and a range of different high-level services. It provides personalisation, single sign-on (SSO), aggregation and customisation features. A so-called 2nd generation portal normally consists of different portlets to process consumer requests to these services and generate dynamic content from the responses. Portlets are used in portals as self-contained pluggable user interface components to the services. Portlets can be developed in different languages and provide accessibility of portlets to the wider international community. Development of portlets in Java is the norm and it adheres to the Java Specification Request 168 Portlet Specification (JSR 168), which standardizes the interoperability of between portlets and portlet containers. JSR 168 compliant portlets are therefore container and framework independent and can be deployed under any portlet container which adheres to JSR 168 specifications. See [7].

3. SERVICE ORIENTED ARCHITECTURE

Modular software is required to avoid failure in large systems, especially where there are complex user requirements. Even in a modular system, inter-dependency of sub-components may make it difficult to meet changing requirements without re-engineering most of the code. This extra effort to maintain software makes re-usability of components difficult if not impossible. Monolithic software with its tightly coupled components is therefore typically specific to its original context and most 1st generation portals were designed in this fashion. The failure of some

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MGC 2005, November 28, 2005, Grenoble, France.
Copyright 2005 ACM 1-58113-000-0/00/0004...\$5.00.

ambitious projects was enough to promote re-designing of software regardless of its complexity, context and size based on independent services. This leads to the concept of a Service Oriented Architecture (SOA). A SOA is essentially a collection of self contained, pluggable, loosely-coupled services which have well-defined interfaces and functionality with little side effect. A service is thus a function that is self-contained and immune to the context or state of other services. These services can communicate with each other either by explicit messages which are descriptive, rather than instructive or there could be a number of “master” services coordinating or aggregating activities, e.g. in a workflow. SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service invokes a unit of work done by a service provider to achieve desired end results for a service consumer. The consumer-provider role is abstract and the precise relationship relies on the context of that specific problem. SOA achieves loose coupling among interacting software agents by employing two architectural constraints: (i) a small set of simple and ubiquitous interfaces to all participating software agents; (ii) the interfaces should be universally available for all providers and consumers. For a fuller discussion of SOAs and classification of possible service components for research, education, collaboration and access to information see [19].

3.1 Requirements for SOA

SOA should be developed to meet the following requirements in one way or another to address problems and issues discussed above:

1. The most important requirement is to leverage existing assets. Strategically, the objective is to build a new architecture that will tactically integrate existing systems, such that over time, they can be componentised or replaced in manageable, incremental projects;
2. Support all required types or “styles” of integration such as:
 - User Interaction - being able to provide a single, interactive user experience achievable through portals and portlets,
 - Application Connectivity - communication layer i.e. middleware that underlies all of the architecture,
 - Process Integration - choreographs applications and services through process model called “workflow”,
 - Data Integration - incorporate data flow within aggregated Grid Service,
 - Portal Integration - provides presentation layer to diverse resources to access them through single location;
3. Architecture should allow incremental implementations and migration of assets. Many integration projects have failed due to their complexity, cost and unworkable implementation schedules;
4. Development environment built around standard component framework such as portal/ portlet specifications, WSRP, Web Services; promote better re-use of modules and systems and allows timely implementation of new technologies;
5. Allow implementation of new computing models; specifically new portal-based client models, Grid computing, scientific research computation and even on-demand computing.

4. PORTAL FRAMEWORK

Software agents which are the building blocks of SOAs are self-contained, which means they should not be modified, but they typically lack any presentation layer. A Portal Framework can provide presentation capabilities for these software agents. The framework is also responsible for providing the required resources and environment for proper functioning of the components plugged into it. The framework is an extra layer in the architecture that provides a standard (presentation) interface for business logic that is independent of programming languages or platforms. At its core, there is a universal API built on the top of the application architecture. Where traditional application development architectures typically have three layers: database, application logic and interface, a Portal Framework has this fourth Presentation Layer that sits between the application logic and the user. The portal not only presents the application logic contained in the software agents but can be used to coordinate different loosely-coupled services into a single concrete service, by providing the related gluing framework.

As mentioned earlier, portals consist of different but related portlets each encapsulating a different self-contained function which may be an aggregation of several base services into a high level service. One big challenge for SOAs is to achieve good scalability, performance and reliability, which is hard because of lack of flexibility to change independent components (although there are injection mechanisms like Spring framework which allow certain modifications). There is always a trade-off between re-usability, implying many fine-grained services each with an overhead, and better performance from course-grained components.

A Portal Framework takes responsibility for message flow from user to service and for inter-portlet communication. The messages can be stateless or stateful, but are normally stateless as software agents are context independent. The framework then either adds state information to the message for multiple interactions per user or stores the state information in a persistent way removing the need for services to maintain state when invoked from different portlets. Any service failure thus does not result in loss of state as the state of services is known. A service providing the software agent can even be replaced dynamically during the execution with another equivalent one. This potentially makes recovery from partial failure relatively easy and services seen by the user can be made reliable.

Traditional stateful services require both the consumer and provider to share the same consumer-specific context, which reduces the overall scalability of the service provider component and increases the coupling between the service provider and consumer making switching of service providers more difficult. Maintaining state through the Portal Framework and aggregating services as portlets is however not a large overhead and the main purpose and benefits of the SOA are then not compromised. The ability to use stateless idempotent services results in less overhead on the service-providing component and uniform behavior when components are used in different ways. These core functionalities in a Portal Framework make it a most appropriate companion to the SOA.

5. IMPLEMENTATION AND STANDARDS

The JSR 168 specification [9] is based on the mature servlet standard following a community review in 2003. The behavior of portlets is similar to that of servlets in many ways, i.e. both portlets and servlets are Java-based Web components, managed by a container, used to generate dynamic content and interact with Web clients via a request/ response paradigm. Unlike servlets, portlets have additional features and limitations, for example, portlets only generate markup fragments and have pre-defined modes and states, but there are optional extensions allowed. Portlets are compatible with J2EE thus providing additional capabilities to typical the SOA architecture. As described above portlets provide persistence to the SOA either through servlet or JSP interacting directly with a database through JDBC, an abstract interface such as Hibernate or using Enterprise Java Beans (EJB). Portlets give new flexibility and extensibility to SOA by enabling the best use of J2EE. JSR 168 allows legacy Web applications to be deployed as portlets in a Portal Framework.

Portlets are not confined to one Portal Framework, based on standard Web services technology OASIS released the Web Services for Remote Portlets (WSRP) also in 2003 aiming to define a standard for interactive, user-facing Web services to make portlets hosted by different geographically distributed Portal Frameworks accessible in a single portal [7, 8]. Unlike traditional Web services however, WSRP defines a protocol which is focused on transferring the markup fragments generated by portlet producers. Although WSRP is still at an early stage as far as implementation is concerned, it indicates the future of portlet/ portal development. Ideally, a deployed service with a portlet interface can be published and consumed in many different portal frameworks. This remote sharing of a single portlet will greatly ease the construction of large-scale portal based systems, or Virtual Research Environment, enabling them to be more scaleable, manageable and maintainable in the long run.

6. NGS PORTAL RELEASE 2.0

The National Grid Service (NGS), funded by JISC, CCLRC and EPSRC, is the core UK grid, which is build up for the production use of computational and data Grid resources [1]. In order to provide users with transparent easy access to these resources, the NGS Portal has been developed and deployed for production usage. The first release of the NGS Portal was based on CHEF [25].

6.1 Adopting Standards

With the goal to deploy a fully-functional portal which could integrate tools coming from other e-Science projects and provide generic tools back for them to re-use, it was decided that a JSR 168 compliant framework was needed. Several such frameworks have been evaluated, including: uPortal, eXo Platform, Liferay, GridSphere and StringBeans [14]. Portlets using the JSR 168 API have been shown to be easily portable between all the frameworks tested. Because of its ease of use, configurability, and support offered by the developers, we are currently using the StringBeans framework release 2.4.2 [15]. A screenshot of the home page of the NGS Portal release 2.0 is shown in Fig. 1.

6.2 Portlets for NGS Portal

Services are provided in the NGS Portal using portlets which can be customized and personalized by the user. Currently available portlets include; MyProxy Proxy manager, MDS information resource browser, GRAM job submission manager, GridFTP browser, Batch job status monitor, Storage Resource Broker (SRB) browser, OGSA-DAI and other collaboration tools are being ported.

Some additional tools use Java Webstart or applets to provide necessary client-side functionality, e.g. for uploading a proxy to a MyProxy server.

The current limitation of portlet technology is that session information cannot be shared between portlets unless all portlets are bundled into a single Web Archive (WAR). All these portlets are therefore included within a single application called the Grid portlet application. Because all included portlets can share session information it is then possible to share Grid proxy credential through sessions.

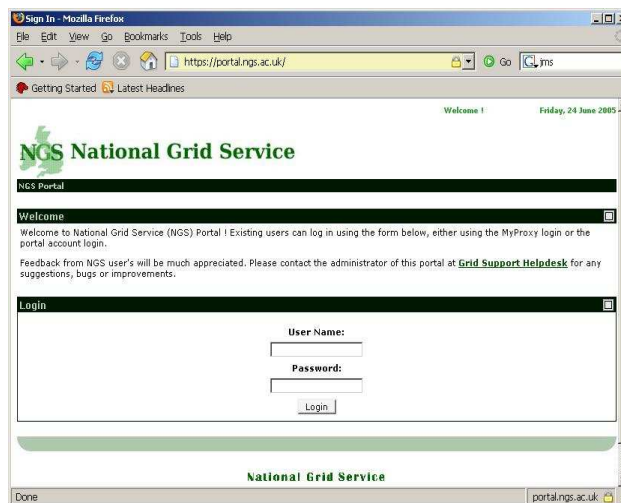


Figure 1: Welcome page of the NGS Portal release 2.0

6.3 Single-Sign-On via MyProxy

The Globus Toolkit uses PKI-based Grid Security Infrastructure (GSI) for Single Sign-On in a Grid environment. It is necessary to retrieve a Grid proxy credential before accessing any Grid resource. A Web-based portal which supports the full Grid functionalities usually follows two steps to retrieve a proxy credential: 1) a user logs in the portal using his/her login details; 2) once authenticated, the user uses a tool within the portal to retrieve a Grid proxy credential commonly through an external MyProxy server. With this proxy credential, it is then possible to access remote resources. In this way, the user has to remember two separate accounts, i.e., a portal account and a MyProxy account. In NGS portal we have integrated SSO using the Java Authentication and Authorisation (JAAS) framework. According to the JAAS architecture, different types of authentication technologies can be plugged into the JAAS framework. StringBeans has the JAAS framework built in and NGS portal makes full use of its capabilities for integrating the MyProxy login mechanism. NGS users can be dynamically logged into the portal by using their MyProxy login details. In fact it is even possible to share information between different login modules,

which is the key feature to achieve single sign-on. As explained, the one-step authentication mechanism brings the benefit of automatically retrieving end users' proxy credentials that are necessary to execute any Grid-related tasks such as submitting a job to a remote computing resource, executing file transfer between local and remote computers. Once the users are authenticated through the MyProxy Login Module, they are mapped to portal accounts automatically. The portal accounts are therefore completely hidden from the view of end users. NGS Portal also provides the conventional way of logging in using username and password and therefore the user can be authenticated in two ways. First the portal tries to authenticate a user through its own record. If authenticated, the user profile will then be loaded and user role(s) will be set. This is the default NGS Portal login process. If the authentication fails logging as a portal user, the user will then be directed to the MyProxy Login Module and will login as a Grid NGS user. Once the user is authenticated, he/she will be mapped to a portal local account and the user's Grid proxy credential will be saved in a separate database rather than modifying the portal's own database. Then the user's role(s) will be set according to the StringBeans record. Once this is performed, the control is then passed to StringBeans. Portlets will be able to retrieve the user's proxy credential automatically from the database.

6.4 Inter-Portlet Communication

A big issue in the current JSR 168 standard is that inter-portlet communication is not supported. As defined in the specification, each portlet application is an independent Web application. It is currently only possible to include all portlets inside one portlet application, as we do in the Grid portlet application, to communicate with each other, for instance, to share the proxy credential in sessions. Some portal framework vendors like IBM with WebSphere [17] provide inter-portlet communication feature at the portlet application level. In fact, there is always a need to communicate at the portlet application level, i.e., to communicate between portlet applications.

The Java Messaging Service (JMS) [18] has been tested in the BatchJobMonitor portlet. Once a job is finished, a JMS message is sent to a JMS server. This message can then be consumed by any JMS client program. This can be further developed as a notification feature, e.g. send an email to the user when the job is completed. The BatchJobMonitor portlet shows all jobs submitted by the user, according to the job status, time, executable, etc.

6.5 Portlet Cloning for e-Science Projects

Looking at existing science portals such as Astrophysics Computing Portal or Mississippi Computational Web Portal, we note that that these application-centric portals actually have many services similar to the NGS. Most e-Science portals in fact provide a set of services such as user profile management, file transfer management, security, etc. One benefit of adopting the JSR 168 standard is that it is possible to make the Java code reusable since JSR 168 defines a set of standard APIs for portlet development. The Grid portlet application we have developed has been successfully deployed and used in other portal projects such as the Integrative Biology (IB) Portal without any software modifications. The initial IB portal uses the GridSphere framework. We are also investigating the use of WSRP, by

which a project may access generic portlets from the NGS portal remotely.

Projects with which we are currently working to share code are: Sakai VRE Demonstrator, Integrative Biology VRE Demonstrator, e-HTPX e-Science projects, SCARF (a cluster on the CCLRC Facilities Grid) and ReDRESS [19].

7. Remote Portlet

The concepts of SOA and portlet cloning is not limited to one specific server or Portal container as just as portlets are not confined to one framework. Web Services for Remote Portlets (WSRP) defines a standard for interactive, user-facing Web services to make portlets hosted by geographically distributed portal frameworks accessible in a single portal. WSRP specifies a protocol focused on transferring the mark-up fragments generated by producers to portal/non-portal and Web/non-Web consumers. This remote sharing of portlets will greatly ease the construction of large-scale portal based systems, or Virtual Research Environments, enabling them to be more scaleable, manageable and maintainable; the core theme of Service Oriented Architecture (SOA). WSRP is suggested to be a natural tool for SOA systems; providing the missing presentation layer with additional needed features in the existing SOA.

WSRP is a cross-vendor protocol that defines a set of interfaces for enabling portal and non-portal Web applications alike to incorporate portlets deployed remotely. It was designed to enable businesses to provide pre-defined interfaces to content or applications in a form that does not require any manual adaptation by consuming applications. WSRP thus enables developers to consume portlets published by other portal sites, regardless of the business system (portal framework) they use. It allows business-level tools to integrate different similar and related portlet services in a dynamic fashion (coupling of services on the fly) by publication and discovery in registries such as UDDI using WSDL [23]. Like any Web service, WSRP is also language agnostic although most tooling currently only exists in Java (the JISC-funded Go-Geo! Portal project has recently written a Perl WSRP module [24]).

7.1 WSRP Extends SOA – Portal Middleware

The main objective of SOA, to aggregate services into a system, is designed to allow for change. The system can use established services or adopt new services that appear after the original services and clients have been deployed, but these must not break functionality. The availability and stability of individual services is critical for the functionality and performance of the system. WSRP provides enhanced and extended support for the following crucial requirements of SOA in a portal context: (i) Secure Login; (ii) Single Sign-On (SSO); (iii) Quick Development; (iv) Component Re-use; (v) Loose-Coupling; (vi) Ease of Configuration and Use; (vii) Statefulness; (viii) Identity Management; (ix) Granularity; (x) Plug-and-Play Architecture; (xi) Flexibility; (xii) Extensibility; (xiii) User Interaction; (xiv) Application Connectivity; (xv) Process Integration; (xvi) Information Integration; (xvii) Build to Integrate.

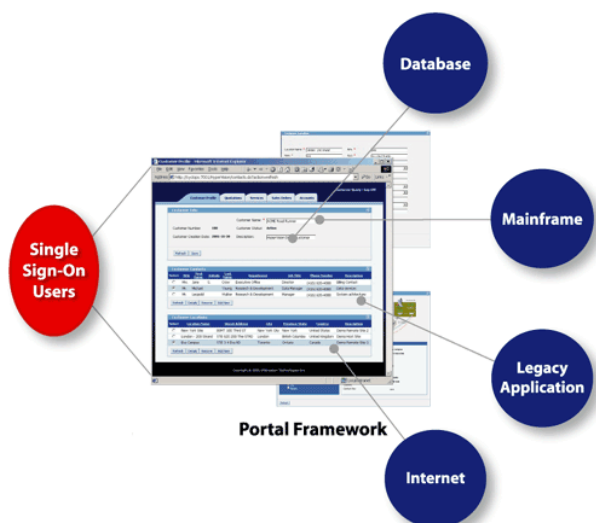


Figure 2 Single Sign-On Portal View

The core of the idea is pretty straightforward - take all your software and make them available as services on the network. Enterprise software is not usually that flexible and it was not built to be used as services so companies have to add service layers to expose existing “heritage” applications for them to be consumed and integrated into other applications. This is a temporary solution that seems to work for now; WSRP provides a natural solution to the demanding challenge of developing sophisticated Web-based/ non Web-based user interfaces through its plug-and-play approach with flexibility to accommodate dynamic requirements. In WSRP the consumer and provider interact via pre-defined message exchanges independent of the content and context of the problem which is key for a scaleable and practical SOA.

7.2 Example Scenario

Web services offer a mechanism to create platform-independent services, and JSR-168 defines a standard by which to develop portlets. While Web services give the ability to re-use back-end services, WSRP enables re-use of the entire user interface.

It might be useful to provide researchers with a portal with the ability to submit jobs on a remote machine. There is likely, already a Web service which provides precisely this capability. We can search for the UDDI registries for a required/ compatible Web service. After discovering such a service we need to code a client to bind to and consume the Web service, as well as to develop a portlet to allow portal users to interact with the service and deploy the newly developed portlet and Web services client on the portal server. At the end of the day, we’ve developed and deployed job submission service functionality to the end-user as a portlet. Re-using the existing services reduces effort and development time. Publishing the front end of this service as a portlet through a WSRP producer allows other researchers to search for and consume it. We need well-maintained registries of WSRP producers where researchers can search for portlets available and use them in custom fashion tailored for their project. Our goal is to make a Virtual Research Environment (VRE) by aggregating existing services and user interfaces in different ways. A project-specific portal can be an aggregation of generic portlets developed by different people and providing user

interfaces to geographically-distributed services creating VRE based on SOA.

8. CONCLUSION AND FUTURE WORK

As discussed in this paper, the emergence of portal technology is providing benefits in developing portlet interfaces to applications to meet the current and future requirements of CCLRC departments and collaborators. Portlets can be re-used by different projects, e.g. the high-profile Integrative Biology project (with University of Oxford), and in different JSR 168 compliant portal frameworks. Deployment and maintenance of applications developed as portlets becomes easier and manageable. A community process is already beginning and many portal frameworks come with useful free-to-use portlets. Because rendering is carried out in the framework, applications can be easily accessible and internationalised. Portlets are compatible with J2EE thus providing additional capabilities required in the SOA architecture. Portals used as a rich client can allow users to customise or personalise their user interfaces and even their workflow and application access. CCLRC facilities will be able to leverage the work so far carried out on the NGS and e-HTPX portals which are fully functional and have received detailed user feedback. This demonstrates the usefulness of providing advanced capabilities for e-Research and having the associated business logic in an SOA loosely coupled from the presentation layer for an Integrated e-Science Environment.

A set of Grid related portlets for UK e-Science users to access the National Grid Service through a user-friendly Web interface has been developed in the Grid Technology Group at CCLRC Daresbury Laboratory. By adopting the JSR 168 standard in release 2.0, the Grid portlet application has been successfully ported to other portal frameworks rather than StringBeans only. It has also been deployed for other portals with minor modifications in portlet application configuration files. The NGS Portal release 2.0 now provides a more standard and flexible way to simplify the usage of the traditional two-step login method to retrieve proxy credentials for end users. The authentication module also illustrates the benefit of adopting JAAS in portal frameworks, which simplifies customizations of portal authentication. With new technologies such as Shibboleth emerging, it is always possible to adopt them by developing new JAAS login modules, which makes the portal framework automatically support the new technologies.

In the future, WSRP will be further investigated to enable the NGS Portal to publish its portlets as remote portlets so that they can be consumed in different project-specific portals. This will greatly ease the construction of complex portal based systems, e.g., Virtual Research Environments (VRE), enabling them to be more scaleable, manageable and maintainable.

Besides JMS tested in the NGS Portal release 2.0, the J2EE architecture integrates a lot of powerful components like Enterprise Java Beans (EJBs) and Web Service support, for developing, deploying and managing multi-tier server-centric enterprise applications. The benefits of using J2EE techniques are currently under investigation.

9. ACKNOWLEDGMENTS

We thank the UK e-Science Core Programme for funding some of the work on the NGS Portal via the Grid Operations and

Support Centre, JISC for funding work on the GROWL and Integrative Biology VRE and Sakai Demonstrator projects via its VRE Programme and EPSRC for funding some work via a grant for Workflow Optimisation Services for e-Science.

10. REFERENCES

- [1] The UK National Grid Service <http://www.ngs.ac.uk/>.
- [2] Globus Toolkit <http://www.globus.org/>
- [3] HotPage Grid Computing Portal, <https://gridport.npaci.edu/>.
- [4] X. Yang, M. Hayes, K. Jenkins and S. Cant, The Cambridge CFD Grid Portal for Large-Scale Distributed CFD Applications, *International Conference on Computational Science 2004 (ICCS2004)*, eds. M. Bubak *et al.*, Springer-Verlag, LNCS 3036, 2004, pp. 478-481.
- [5] A.J. Richards, R.J. Allan, G. Drinkwater and K. Kleese, Building the e-Science Grid in the UK: Advanced Grid Computing, *UK e-Science AHM2003*, Nottingham, UK, 2003.
- [6] R.J. Allan, D. Baker, D. Boyd, D. Chohan, S. Cox, H. Eres, R. Fowler, N. Furmento, J. Giddy, T. Harmer, M. Hayes, N. Hill, J. Hillier, J. Jensen, A. Keane, M. Krznaric, W. Lee, M. McKeown, A. Mills, S. Newhouse, S. Pickles, R. Pinning, A. Richards, A. Saleem and J. Watt, Building the e-Science Grid in the UK: Middleware, Applications and Tools Deployed at Levae 2, *UK e-Science AHM2003*, Nottingham, UK, 2003.
- [7] R.J. Allan, C. Awre, M. Baker and A. Fish, *Portals and Portlet 2003 NeSC Workshop*, 14-17 July 2003, http://www.nesc.ac.uk/technical_papers/UkeS-2004-06.pdf.
- [8] Web Services for Remote Portlets (WSRP), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp.
- [9] JSR 168: Portlet Specification, <http://www.jcp.org/en/jsr/detail?id=168>.
- [10] The Sakai Project, <https://www.sakaiproject.org/>.
- [11] Open Grid Computing Environment, <http://www.collab-ogce.org/nmi/index.jsp>.
- [12] Globus Java Commodity Grid Kits, <http://www.globus.org/cog/>.
- [13] J. Novotny, S. Tuecke and V. Welch, An Online Credential Repository for the Grid: MyProxy, *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, pp. 104-111, 2001.
- [14] A. Akram, D. Chohan, X.D. Wang, X. Yang and R. Allan, A Service Oriented Architecture for Portals Using Portlets, *UK e-Science AHM2005*, Nottingham, UK, 2005, accepted.
- [15] StringBeans Portal, <http://www.nabh.com/projects/sbportal>.
- [16] Java Authentication and Authorisation Service, <http://java.sun.com/products/jaas/>
- [17] WebSphere, <http://www-306.ibm.com/software/websphere/>.
- [18] Java Messaging Service (JMS), <http://java.sun.com/products/jms/>.
- [19] R. Crouchley, A. Fish, R.J. Allan and D. Chohan, Virtual Research in the UK: Portal Services for Awareness and Training in e-Science, *UK e-Science AHM2004*, Nottingham, UK, 2004.
- [20] Akram, A., Allan, R., and Crouchley, R. WSRP reincarnation of service oriented architecture, *UK e-Science AHM2005*, Nottingham, UK, September 2005, accepted.
- [21] Akram, A., Chohan, D., Wang, X.D., Meredith, D., and Allan, R. CCLRC portal infrastructure to support research facilities, *GGF Workshop on Science Gateways, GGF14*, Chicago, IL, USA, June 2005.
- [22] North West Grid, <http://www.nwgrid.co.uk/>.
- [23] Universal Description, Discovery and Integration (UDDI), <http://www.uddi.org/>
- [24] Go-Geo! Portal Perl modules for WSRP <http://www.gogeo.ac.uk/PortletInfo.html>
- [25] CHEF: *CompreHensive collaborativE Framework* originally developed at the University of Michigan.