CCLRC Portal Infrastructure to support Research Facilities

Asif Akram, Dharmesh Chohan, Xiao Dong Wang, David Meredith and Rob Allan
e-Science Centre, CCLRC Daresbury Laboratory

1. Introduction

The CCLRC e-Science Centre is working on a number of UK e-Research projects supporting access to large research facilities such as the neutron spallation source (ISIS), the Synchrotron Radiation Source (SRS) and the Central Laser Facility (CLF) and the new synchroton light source, Diamond in addition to the National Grid Service and the HPCx supercomputing facility. The aim of this work is to create an Integrated e-Science Environment for CCLRC. A clear requirement of facility users is to provide seamless access and integration of these resources. This can be done by developing portal interfaces for each facility or project exposing their services as portlets for reuse between the different portal frameworks and CCLRC projects. This paper outlines the benefit of using portal frameworks and portlets to meet this goal with a Service Oriented Architecture designed to complement desktop tools. An example is given of the portal services being provided for the National Grid Service (NGS) and e-HTPX portals, which include computational and data Grid production resources for research.

2. Portal Frameworks, Portlets and Standards

Portal frameworks and Portlets are relatively new 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 used to process consumer requests to these services and generate dynamic content from the responses. Multiple portlets can be hosted within a single portal as self contained pluggable user interface components (i.e. providing aggregation of different functionality). Portlet interfaces can be developed in different languages but CCLRC project portal interfaces are mainly based on Java technology and managed by a Java portlet container. Java portlets adhere to the Java Specification Request 168 Portlet Specification (JSR 168), which standardises the interoperability of portlets between different portlet containers. JSR 168 compliant portlets are therefore container and framework independent and can be deployed within any container that adheres to JSR 168 specifications. See [1] & [5]. Portlets are not confined to one portal framework. Based on standard Web services technology OASIS [4], released the Web Services for Remote Portlets (WSRP) also in 2003. WSRP aims 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 [1].

3. Single Sign On

Single Sign-On (SSO) is a key requirement in providing Authentication and Authorisation services to access facilities. We are evaluating technology to determine a security framework which is easily scalable in terms of extending the framework to different user account management systems such as LDAP, Active Directory Service or MyProxy server using X.509 certificates. The aim is to make the user login experience independent of the authentication technology. In the NGS portal we are working to integrate the MyProxy server authentication mechanism to access the NGS portal. This will be achieved by extending the Java Authentication and Authorisation Service (JAAS) [6]. JAAS implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization. Some portal frameworks support JAAS, e.g. StringBeans, which is used by NGS Portal. Authorisation of an NGS user will be accomplished by using JAAS and portal framework role-based access control functionality. JAAS authorization extends the existing Java security architecture that uses a security policy to specify what access rights are granted to executing code. We are also looking to evaluate another Single Sign On framework called Java Open Single Sign-on (JOSSO) [7]. Some main features of JOSSO include strong authentication using X.509 client certificates,

support for multiple simultaneous authentication systems, integration of Java and non-Java applications and a security model based on JAAS, SOAP Web services, EJB, servlet/ JSP and Struts standards. These technologies will provide links into the UK e-Science Certification Authority and CCLRC's Corporate Data Repository.

4. Desktop Clients to Access Grid Resources

The latest trend of application development shows the popularity of Web applications among users and developers. Web applications provide uniform and pervasive interfaces to remote services and resources regardless of user environment. For developers they are easy to develop and maintain and for users there are no installation and configuration issues. Today Web applications are very mature courtesy of standards like HTML, JavaScript, CSS and XML related technologies but are still lacking in features compared to desktop applications. Desktop and Web applications are complementary and address different sets of requirements, although there is overlap in basic functionalities. For simple distributed resources where user interaction is limited, Web applications are the better choice, but for Grid services where user interaction is more complex, desktop applications may be a better choice. We are aiming to support both desktop and Web interfaces considering the user and project requirements through the use of Web services.

Development is carried out in a modular way to avoid potential failure in large systems, especially where there are complex user requirements. The concept of SOA was developed to avoid the problems of having to re-write large portions of code to accommodate changing requirements. A SOA is essentially a collection of autonomous, self contained, pluggable, loosely coupled services that 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. These core and atomic services can be integrated into either Desktop or Web applications (scaled down version) for uniform user experience.

The Workflow Optimisation Service for E-Science (WOSE) project started recently with CCLRC, Imperial College and University of Cardiff. The purpose of this project is to investigate optimisation strategies for workflow execution for Web services; (1) using Business Process Execution Language (BPEL), see [9]; (2) Simple Conceptual Unified Flow Language (SCUFL), see [10] and (3) Business Process Modelling Language (BPML), see [11]. The idea is to develop workflows from the point of view of users with limited knowledge of workflow languages.

In WOSE, dynamic Web service invocation and interoperation between different workflow languages and workflow engines will be used. We aim to enhance the user experience and provide links for known Web services and for discovery from private or public Universal Description, Discovery & Integration (UDDI) registries. For ease in engineering workflows, we aim to develop a drag-and-drop application. In this case, a robust, dedicated "fat" client is better than a browser-based client. If instead, the client software is used by a large number of people requiring only limited functionality, a Web client is ideal. Browser based clients have an inherent advantage as they do not need to be upgraded on the client side and provide a pervasive access mechanism.

Considering the user requirements we can identify two categories: (1) users executing existing workflow for complex scientific procedures in which they may not be experts, and (2) expert users engineering new or existing processes as workflows. For the first type of users Web Applications are appropriate since executing existing workflows means uploading the workflow and supporting parameters and constraints to the server for use by the workflow engine. The second type of user needs rich client software with advanced functionality. This includes comfort features, such as a polished user interface, drag-and-drop, and a rich set of keyboard shortcuts, and provision to save partial workflows and integrate them as building blocks for complicated and sophisticated workflows.

4.1 Practical Issues

In real world workflow scenarios, workflows are not simple tools that call different Web services in fixed static or dynamic manners. Rather, a workflow engine or workflow developer has to tailor the output from one Web service according to input requirements of the next Web services. This transformation can be simple and automatic or can be complicated involving creation of new data structures (via an XML Schema) based on the output of previous activities. Things become more complicated when a newly created data structure requires input values from a user

during the execution of the workflow. There is no option in current specification to "pause" workflow, notify users to input required parameters and resume the workflow after input. Moreover, Web services orchestrated through workflow are deployed on geographically distributed servers, which may have security and access or service policies, which should be negotiated before execution of the services. Workflow specifications don't have any provision to support these additional policy constraints and there should be a mechanism to plug-in policy parameters in the workflow. Static policy constraints can be kept in the workflow as variables but if they are dynamic and are based on the output of previous activities, user interaction is required. Grid resources are more dynamic in nature and workflows using them are not predictable.

Based on different usage scenarios, we have designed a set of "integration services" to extend existing workflow specification BPEL. These services constitute CPU resource overhead and are suitable for Desktop applications rather than Web Applications. Integration services developed or planned for workflow extensions include: (1) Data conversion or Data Structure creation using XSLT; (2) Messaging services to inform the user of workflow about an unexpected result/fault during execution (meanwhile pausing the workflow execution for further instructions); (3) Maintaining a pool of similar/ compatible Web Services, failure or un-availability of Grid resource will not result in premature termination of workflow and dynamically other compatible Grid resource will be tried based on rating mechanism discussed later. If a system can't find any other compatible service the workflow will have to be paused and resumed later, either automatically based on predefined constraints or with intervention of the user; (4) Integration of local java classes with the workflow in pre-defined manner, which may add Soap Headers, Digital Certificates for security, Web Service Policy based constraints/ parameters before calling the Web Service. This information is not usually available in WSDL documents and client and service providers should have negotiated and agreed before using/ offering the service. A workflow engine has no way to adjust itself at run time and without these constraints and additional information workflow can't call Web Services successfully; (5) A rating mechanism to rank similar Web services in a pool based on different policies which are configurable and can modified according to different SLA requirements. The rating system will help dynamical selection of the best available service from pool of services; (6) A graphical monitoring tool to monitor the workflow in scientific services when the workflow cycle is lasting for days if not weeks. It will be nice to have user interaction with a monitoring tool to stop/ pause/ resume/ cancel/ modify the workflow at run time giving more flexibility to the scientist; (7) Information Persistence is supposed to store information at different stages (breakpoints) of workflow which includes variables or java object serializations to re-run the workflow between different breakpoints on demand. Many of these are similar to the integration services provided by a portal on the server side.

5. Portals and Web Service Examples Developed at CCLRC

We give a couple of examples which substantiate our work.

5.1 The UK National Grid Service

Services are provided in the NGS Portal using portlets which can be customised by the user. Stringbeans is used as the framework. Currently available JSR 168 compliant portlets include; MyProxy management, MDS resource discovery, GRAM job submission, GridFTP, Job status monitor, SRB, OGSA-DAI and collaboration tools being ported. As an example of this work, we note that OGSA-DAI is a core service of the Globus toolkit developed by the Globus Alliance at University of Edinburgh. It provides an extensible middleware Grid framework for easily integrating data from independent data resources and has the following features; allows data access, allows data discovery, facilitates data integration, and provides a uniform interface to access data.

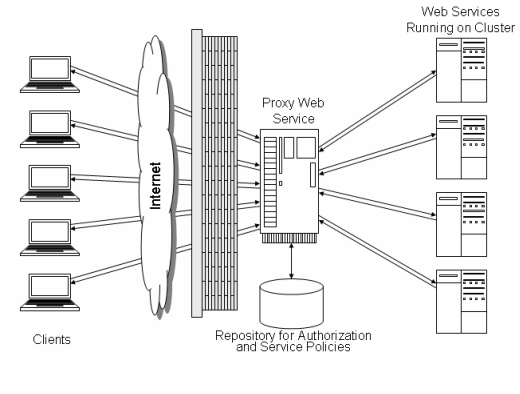
We have developed a portlet interacting with OGSA-DAI Web services using Tomcat and Axis SOAP implementation. OGSA-DAI can be used with or without a security model. The current implementation of OGSA-DAI deployed uses a Message Level Security to access data. Further work will be carried out to provide access to OGSA-DAI services using Authentication and Authorisation mechanisms mentioned above. This, and other work on OGSA Grid services, was started in the UK OGSA testbed project with CCLRC and the Universities of Portsmouth, Manchester and Reading. Some additional tools use Java Webstart or applets to provide necessary client-side functionality, e.g. for uploading a proxy or accessing desktop files.

5.2 Web Services and Portals in e-HTPX

e-HTPX is a BBSRC (Biotechnology and Biological Sciences Research Council) funded UK e-Science project for structural biology [3]. The project aims to develop both a communications infrastructure and a suite of related user interfaces designed to allow the planning and remote execution of protein crystallography experiments. The project integrates a number of key services provided by UK e-Science, protein manufacture and synchrotron radiation facilities (Diamond, ESRF). The project implements and tests a number of distributed computing technologies, including those discussed above (especially portal development, communications via Web services and provision of 'gateway' style Web services to provide access to service-endpoints situated inside an institutional firewall).

The e-HTPX portal is a distributable Web application that provides both an interface to input necessary data, and a single point of access to the underlying e-HTPX Web service framework. The portal acts as both a Web service client and a Web service-response hub for collation of responses from different services (example services situated at the synchrotron site include those associated with X-ray data collection and HPC services designed to aid the determination of protein structures). Two portal architectures have been designed in order to suit different research environment needs: a) the service-site portal, which is maintained by each synchrotron, and offers a standard service for the majority of clients; and b) the client-site portal, which can be installed at a client institution and allows configuration and close integration of the portal functionality with a client institutional LIMS (Laboratory Information Management System). Despite the differences in portal architecture, both portal implementations act as clients to the services hosted at the synchrotron. When using the portal to submit Web service requests to the synchrotron, authentication is first implemented by the portal application layer (i.e. when logging onto the portal interface). The security credentials are also passed to the synchrotron web services allowing secondary (site-specific) authentication and authorization.

Figure 1: Proxy Web Service for Authorisation



With regard to the communication infrastructure at the synchrotron site, it has to be anticipated that firewall policies will limit direct access from external clients to the key HPC and data collection services (i.e. it is most likely that these resources will be located inside internal firewalls, and can only be accessed via known gateway servers). Consequently, an extensible 'gateway' style Web service framework has been designed for accessing these resources. The gateway model means that all client requests are sent to an externally visible 'gateway' Web service before being routed to the ultimate receiver (the requested service). The client can't submit a request directly to the service and knows nothing of the service endpoint (i.e. the endpoint URL). In addition, firewall administrators may implement additional security measures such as IP-recognition between gateway server and service endpoint. This model represents an end-to-end communication involving three 'actors' or 'roles' (client, intermediary, and ultimate receiver), not a point-to-point communication that implements direct communication between the client and ultimate receiver. The role of the intermediary gateway Web service is to perform authorization and authentication, determine the requested resource by parsing the XML payload, parse the input parameters for the requested service, and re-route the SOAP request (including attachments) to the endpoint service. In performing these actions, the service endpoint is ensured to receive properly validated requests from authorized users.

5.2 Final comment on Portlets and WSRP

Currently there are several open source portal frameworks which support WSRP functionality, e.g. uPortal and eXo Platform, but the WSRP is integrated as an internal mechanism. Other portal frameworks which do not support WSRP therefore can not use the functionality directly. We have also developed a generic WSRP consumer portlet using the WSRP4J API [8]. At a later date the WSRP portlet will also support the WSRP

producer functionality. The WSRP portlet will implement presentation oriented service, interaction and portability. The benefit of using the WSRP consumer portlet would be that it can access multiple remote portlets via the WSRP portlet producers. The advantage is that portlets developed as WSRP producers can be hosted on a single machine but can be consumed from any portal environment and hence deployment and maintenance of the Web application can be much easier. This leverages on the SOA concept. This is ongoing work and is particularly of interest in advanced training for scientists where content can be aggregated and combined with collaboration tools, for instance in the ReDRESS project which is a collaboration of CCLRC and University of Lancaster [2]. WSRP may also be a convenient way for commercial tools to be exposed in portals alongside open-source tools. We note that the Sakai project in the USA is currently evaluating such possibilities for Collaborative e-Learning and we are using this in a VRE Sakai demonstrator project, collaboration between CCLRC and the Universities of Lancaster, Oxford and Portsmouth.

6. Summary

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 facilities support. 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 free-to-use useful 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.

7. References

[1] R.J. Allan, C. Awre, M. Baker and A. Fish *Portals and Portlets 2003.* Proc. NeSC Workshop 14-17/7/2003 (CCLRC and NeSC, 2004) http://www.nesc.ac.uk/technical_papers/UKeS-2004-06.pdf

[2] R. Crouchley, A. Fish, R.J. Allan and D. Chohan *Portal Services for Awareness and Training*. Proc. AHM'2004 (Nottingham, September 2004) Paper 223

[3] Rob Allan, Ronan Keegan and D. Meredith *e-HTPX – HPC, Grid and Web-Portal Technologies in High Throughput Protein Crystallography* AHM'2004 (Nottingham, September 2004) Paper 101

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

[5] *Portal specification JSR 168 and API* http://www.jcp.org/aboutJava/communityprocess/review/jsr168/

[6] *JAAS* http://java.sun.com/products/jaas/

[7] *JOSSO* http://www.josso.org/

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

[9] *BPEL, version 1.0, 31 July 200* http://www.ibm.com/developerworks/library/ws-bpel/

[10] *SCUFL* http://taverna.sourceforge.net/docs/beta8workbench/workbench.html

[11] BPML March 8, 2001 http://www.bpmi.org/BPML.htm