A Service Oriented Architecture for Portals Using Portlets

Asif Akram, Dharmesh Chohan, Xiao Dong Wang, Xiaobo Yang and **Rob Allan**

CCLRC e-Science Centre, CCLRC Daresbury Laboratory
Warrington WA4 4AD, UK

Abstract

Portals and Portlets are emerging technologies and gaining lot of popularity. Portals are gaining attention among programmers due to their ease in development, richness in functionality, customization of interface and pluggable architecture. With this popularity today there are many open source Portal Frameworks available and list of these open source frameworks is all the time increasing. It is important to evaluate all these Portal Frameworks in an effective manner, based on core functionality i.e. their compliance with JSR 168 Portal API and optional features available to the programmers i.e. IDE plug-ins, utility packages, monitoring tools etc. We have selected a small number of Portal Frameworks based on their popularity and our experience of using them to evaluate their core and optional functionalities. This paper will outline our findings and current trends in the feature rich Portal Frameworks. Paper will explain our criteria for rating different Platforms and then will discuss each selected Platform. In the last section we have Inter-Platform WSRP compliance test results.

1. Introduction

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 but this paper focuses on portlets based on Java technology and managed by a Java portlet container, which is the norm. Java portlets adhere to the Java Specification Request 168 Portlet Specification (JSR 168), which standardises 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 [1].

2. 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 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 [2].

3. 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 behaviour when components are used in different ways. These core functionalities in a Portal Framework make it a most appropriate companion to the SOA.

4. Implementation and Standards

The JSR 168 specification [3] is based on the mature servlet standard following a community review in 2003. The behaviour 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, Ref. [4], 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, see [1]. 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 portals/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.

5. Evaluation Criteria

It is bit difficult to compare different Portal Framework as each of them addresses different requirements and technologies. We tried our best to be objective and compare different Portal Frameworks with a broad range of criteria to accommodate the speciality of each Portal Framework and maximum consideration of user requirements. In the end, the following criteria were used to evaluate the open source Portal Frameworks mentioned in this paper. These criteria are based on core and optional requirements from Portal Framework and are listed in order of perceived importance. Each Portal Framework has been given a score of 1 to 5 against each criterion, 5 being the most effective. The total score of each Portal Framework is shown at the end of the tabular comparison, with a visual Bar Graph in Section 7.

(i) JSR 168 compliant: It was felt that JSR 168 is the most important requirement for portal development to free developers from vendor specific Portal API and promote re-usability.

(ii) Ease of installation: Portals are meant to be a presentation layer to existing business logic and should not bring complexity to the solution. Here we evaluated the installation process by efforts required to get started, e.g. configuring a database, whether the framework includes

built-in Web container or not? Most of the Portal Frameworks use the Tomcat container and are easy to install, they come bundled with Tomcat or a compatible Web Archive (WAR) file.

(iii) Documentation Standard: Portal Development is similar to Web Application development i.e. Servlet and JSP, but still there are some Portal Framework dependent tweaks to make portals work. Documentation of the Portal Framework with well written examples is important. This criterion examines the completeness, relevance and quality of each Portal Framework's documentation covering both Administration and/or User Guides.

(iv) Online Support: Documentation doesn't answer or addresses all programmers' questions and occasionally (in fact frequently) a programmer may need support from the framework developers. In this criterion we have examined the quality, quickness and appropriateness of the developers' responses to queries. This also includes maintenance of Wiki, flexibility to support new features on demand.

(v) Portal Management: Deploying portlets in the Portal Framework requires configuration of different deployment descriptors, some of them are part of the Portal API, i.e. portlet.xml and J2EE requirements, i.e. web.xml and the rest of them are specific to the Portal Framework. This criterion includes administrative functions, i.e. adding users, assigning roles to users, assigning categories to portlets etc. and user functionality to customise the portlets to meet individual requirements. i.e. layout, skin, adding and deleting portlets etc.

(vi) Portlet Resources: Most of the Portal Frameworks either come with re-usable utility portlets or a repository of portlets such as Mail Portlet, Calendar Portlet, and Search Portlet. In this criterion we have examined the usefulness and re-usability of these portlets supplied with the Portal Framework.

(vii) Performance and Scalability: Architectural design of a Portal Framework is crucial for its performance. Portals are an additional layer to SOA and thus have the capability to slow down the perceived performance of the services. Providing basic portal functionality is not enough in a commercial environment where performance and saleability is also crucial. This criterion examines the performance of Portal Frameworks in terms of start up time; portlets load time, database access time, etc.

(viii) Security: Most of the Portal Frameworks come with default security of user login with password. This default mechanism of authorization and authentication is not enough in commercial or e-Science projects. In this criterion we have examined additional security capabilities of the Portal Frameworks such as Java Authentication and Authorization Service (JAAS) [5], Java Open Single Sign-On (JOSSO) [6] and configuration with SSL.

(ix) Technology Used: Different Portal Frameworks use different optional technologies for the benefit of programmers which are not part of portlet API. In this criterion we have evaluated different popular technologies used by different Portal Frameworks such as Struts [7], Java Server Faces (JSF) [8], Spring [9], Hibernate [10], Tiles [11], Enterprise Java Bean (EJB) [12], and Web Services [13].

(x) Portal Features: Portal Frameworks are not only the portal/ portlet containers hosting different portlets. In fact most of the Portal Frameworks come with additional functionalities to develop real-life J2EE portals such as Content Management System (CMS) [14], Workflow, Administrator Management tools, Framework Monitoring tools. This criterion will examine the optional features available in the Portal Framework and their standard and usability.

(xi) Server Dependency: the portlet API is an extension of the servlet API and doesn't need advanced J2EE features, but in real life most Web Applications are J2EE applications using EJBs for persistence. Web Services can be used to encapsulate legacy code, transactions etc; thus Portal Frameworks are not only confined to the servlet container like Tomcat. It is useful if Portal Frameworks can be deployed in variety of servers and in this criterion we have examined the compatibility of the Portal Frameworks with different open source and commercial servers.

(xii) WSRP Standard Compliant: the portlet API is the presentation layer for the Web Application, but it is not necessary that clients should only be Web based; alternative desktop clients are desirable [15]. The Web Services for Remote Portlets (WSRP) [4] specification makes it possible to consume portals/portlets in non-Web applications such as Java Swing Consumer. In this criterion we have examined WSRP support in the frameworks either as WSRP consumer, producer or both.

6. Selected Portal Frameworks

Portals are gaining attention among programmers due to their ease in development, richness in functionality, customization of interface and pluggable architecture. With this popularity today there are many open source Portal Frameworks available and list of these is all the time increasing. It is not possible to evaluate all these Portal Frameworks in an effective manner and we have thus have to select a small number based on their popularity and our experience of using them for development work. This doesn't mean that other frameworks are below standard or have limitations. Some other commercial and open-source Portal Frameworks are listed in [1]. We have selected the following Portal Frameworks for evaluation:

- Sakai 1.5 (Due to its broad use in Virtual Research Environment (VRE)). See separate report [16].
- uPortal (Due to its tremendous use in Academic Institutes work wide)
- GridShpere (One of the first JSR 168 compliant open source European Portal Frameworks)
- eXo Platform (Due to its popularity)
- Liferay (Due to its popularity, user interface and optional functionality)
- Stringbeans (Due to its ease in use)

- **6.1 Sakai**

The Sakai Project [17] in the USA grew out of the CHEF portal-based Collaborative Learning Environment activity in the USA. In Sakai the CHEF framework has been substantially re-written and the course management and assessment tools developed to be shared by a number of major US teaching institutions. CHEF and Sakai also contain interesting collaboration tools and have been shown to be useful in a research context when augmented with additional Grid and information management tools. It is for these reasons that we carried out the original Sakai Evaluation Exercise [18] on behalf of JISC and are now developing additional tools perceived to be of relevance to e-Research.

Sakai is a large Java-based project using a number of standards such as Hibernate [10] (for persistence and database access), Spring [9] (for code injection and late binding) and Java Server Faces [8] (for the presentation layer). It is however not using JSR 168 as the developers found that it did not support their requirements of full integration of the tool interfaces into the Sakai kernel. Iframes are currently used instead of JSP 168. The kernel itself has been designed along similar lines to JBOSS, but is relatively lightweight and provides a set of essential services used by all the tools. It was found necessary to maintain all the code in a single Tomcat Web Application for persistent session management. The OKI OSIDs are used to guide the design of APIs to the individual tools.

Sakai v2.0 was released on 15th June 2005, but the results in this paper are based on experiences with v1.5. In the future Sakai will offer support for export of WSRP via a thin WSRP4J layer integrated above the kernel. It will then be possible to embed Sakai into uPortal v3.0 towards the end of 2005.

Issues with Sakai, in addition to the lack of a JSR 168 interface, include the variable support for database back ends. Whilst Hibernate is used internally and there is good support for Oracle, much of the SQL is hard-wired and we were unable to get it to work with PostgreSQL. Another issue is that because the Sakai architecture code base is still evolving rapidly it is hard to find reliable documentation and to migrate tools and data from one release to the next in an easy way.

From the users' point of view, the most appealing feature of Sakai is its group-based structure. Each group has its own "worksite" within the portal which is secure from others. Users belonging to more than one group can see an "aggregated" view in their own "personal worksite". Groups can also be set up on-the-fly for specific purposes and be published and maid "joinable". Groups can customise the tools they see from the available pool to meet their own purposes. In an educational and community portal this is particularly important and appears to be quite scalable in Sakai.

- **6.2 uPortal**

uPortal [19] is a widely used Portal Framework in academic institutes and it mainly addresses the requirements of those organisations. uPortal is a very stable Portal Framework and was released even before the JSR 168 specification, due to which uPortal has applied non-standard mechanisms, which they call channels. uPortal is JSR 168 compliant but still most of the features available in the uPortal are based on custom and home grown solution with channel adaptors rather than native portlets. uPortal supports portlets via the Pluto Portal Framework [20]. uPortal is the only open source Portal Framework which supports maximum type of portals ranging from Java portals to HTML portals, text portals to XML portals.

Figure 1: Support for different types of Portals in uPortal

uPortal is widely used in academic institutes due to its built-in support for these institutes and their requirements. uPortal can be used with Central Authentication Service (CAS) [21] to control the access to CASified applications, based on "when", "who", "from where", "what service", in conjunction with LDAP attributes and so on. This kind of "Central Authorization Scheme" is quite powerful for heterogeneous environment like universities/colleges. It is very easy to configure the groups and permissions services which are crucial requirement of University environment in uPortal with the local source of information, e.g., LDAP.

uPortal supports JSR 168 compliant portals/portlets through adapter, and requires standard configuration files i.e. portlet.xml and web.xml. uPortal comes with utility Ant build file which examine and validate the portlet and deployed it in the uPortal Framework.

Documentation of uPortal is not considered to be good and is not up to date. Most of the tutorials for uPortal are written by students doing their theses and are based on the version available at that time so can't be used directly for latest releases. Documentation related to uPortal is scattered about in several places; the uPortal Web site, a confluence-based Wiki, email lists, the uPortal issue management system (using Jira), and external sources. For beginners it is very difficult to search the related information and it will be nice if all information and tutorials can be grouped together at one place.

uPortal supports partial specification of WSRP, and uPortal can be only used as a WSRP consumer with the WSRP4J reference implementation. We tested this partial WSRP implementation and find it is stable, and it works in consistent manner with Remote Portlets of enough complexity. The documentation related to WSRP is virtually none existent and the best information related to uPortal and WSRP is available from external sources such as Oxford University. uPortal v3.0 is in the pipeline and some of these issues will be addressed. The current stable version of uPortal is 2.5.

- **6.3 GridSphere**

GridSphere [22] is a very stable Portal Framework initially funded by the EU GridLab project spanning 3 years from year 2002. GridSphere has very impressive user interface and it is very easy to use. GridSphere is 100% JSR 168 compliant since the first quarter of year 2004. GridSphere's functional portal prototype is based on Portal API borrowed from WebSphere API, which makes it fully compatible with IBM's WebSphere® 4.2. Portlet customisation and personalisation is very simple and is Web interface based and requires no complex configuration and deployment descriptors. GridSphere comes with its on API and higher-level model for building complex portlets using visual beans and the GridSphere User Interface (UI) tag library. This tag library makes portlet development very simple but then portlets are not JSR compliant, which is similar to Liferay Enterprise which provides its own utility package. GridSphere is mainly deployable with Tomcat and with little configuration it can be deployed on fully J2EE compliant server like JBoss, but there is no documentation related to this, and only information available is from user mail list. GridSphere is not supporting many latest technologies or at least not mentioned and claimed by GridSphere team like Struts [7], Tiles [11], Spring [9] but they are supporting portlet development using JSF [8]. GridSphere supports persistence of data provided using Hibernate [10] JDO/OQL for database support, which means any JDBC compliant database, can be used with it without any complicated configuration and modification of code. The biggest portlet feature available for programmers is Integrated Junit/Cactus unit tests for complete server side testing of portlet services including the generation of test reports. GridSphere supports different type of authentication mechanisms, by default it is username and password. It also supports Role Based Access Control (RBAC) to separate users into guests, users, administrators and super users. GridSphere has flexible support for customisation and presentation which is XML based portal presentation description that can be easily modified to create customised portal layouts.

GridSphere has well managed website, with updated tutorials, information, news and all other required information. GridSphere also comes with many utility portlets and they maintain portlet repository. GridSphere has released latest version of Portal Framework 2.0.3 in June 2005.

- **6.4 eXo Platform**

The eXo Platform [23] is defined as a portal and content management system (CMS) [14]. Current version of eXo Platform is 1.0 which was released on Febrruary 2005. Typically eXo Platform has been used as a corporate portal; eXo provides users with a customisable single point of access to the company's information systems and resources. Through the Web environment, eXo provides business information to the firm's employees, allows the exchange and management of its data, as well as the execution of critical business processes.

eXo is a JSR 168 compliant enterprise portal built from several modules. It is based on JSF, Pico Container, JBossMX and AspectJ. WSRP is also supported in eXo. eXo Platform supports different technologies by implementing different bridges. Fig. 2 gives an overview of eXo Platform.

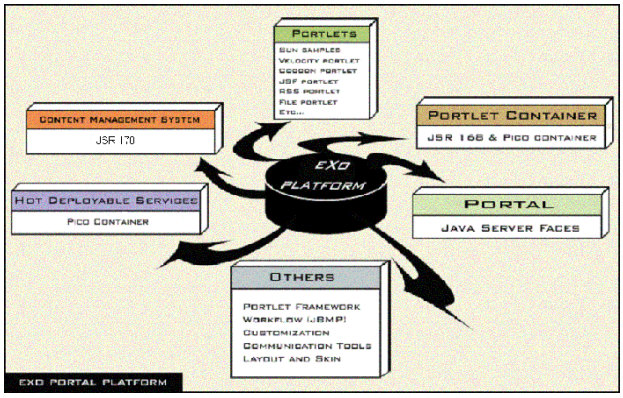

Figure 2: Overview of the eXo Platform, source [23]

The development of the Grid portlet applications for the NGS Portal release 2.0 [24] was mostly based on the eXo Platform because of its support for hot-deployment of portlet applications.

The eXo Platform comes with two pre-compiled distributions, the "express" and the "enterprise" editions. There is in fact no difference between the two editions in terms of functionality and features, but the flavours are for different type of container i.e. Servlet Container and EJB Container. The express edition is to be deployed inside a servlet engine whist the enterprise edition is to be deployed inside a full J2EE 1.3+ application server. Both editions have been successfully deployed under Tomcat 5.0 and JBoss 4.01sp1 accordingly.

Like other Portal Frameworks, there are sets of portlets coming with the eXo Platform, e.g., Web-related, communication-related, content-related, navigation-related, user/admin-related and MVC-related portlets. Workflow and WSRP-related portlets are also included.

eXo brings a layer (Struts bridge) between the portal and any existing Struts application within the portlet, existing Struts applications can be embedded in a portlet with a minimal amount of change. Another bridge, the Cocoon bridge is also included in eXo to embed existing cocoon applications in a portlet fragment.

The support of WSRP in eXo is reported in the next section. At the time this paper was written, WSRP support seems to be limited in eXo.

In general, the eXo Platform is a powerful open-source Portal Framework with many cutting-edge technologies supported. eXo Platform was found best in performance with minimum portal upload time. eXo also comes with eclipse plug-in, which is very convenient for developers and can be extended and tailored to accommodate custom requirements.

- **6.5 Liferay**

The Liferay Portal Enterprise [25] is more than just a portal container; which comes with lot of helpful features like Content Management System (CMS) [14], WSRP compliant producer and consumer, Single Sign On (SSO), support for Aspect-Oriented Programming (AOP), and many other latest technologies. Liferay has a very clean architectural design based on best practises of J2EE, which allows it to be used with a variety of containers ranging from lightweight servlet containers like Tomcat and Jetty, to fully fledged J2EE-compliant servers like Borland ES, JBoss, JOnAS, JRun, Oracle9iAS, Orion, Pramati, RexIP, Sun JSAS, WebLogic, and WebSphere. In fact Liferay is the only open source portal container which supports nearly all commonly used open source and commercial Java Servers.

Flexibility in design allows implementation of business logic in any suitable and appropriate technology like Struts [7], Tiles [11], Spring [9] and EJB [12] which in turn can be based on Hibernate [10], Java Messaging Service (JMS) [26], JavaMail and Web Services. Liferay makes it possible to give Portal Presentation to any type of Java application with no or minimum changes.

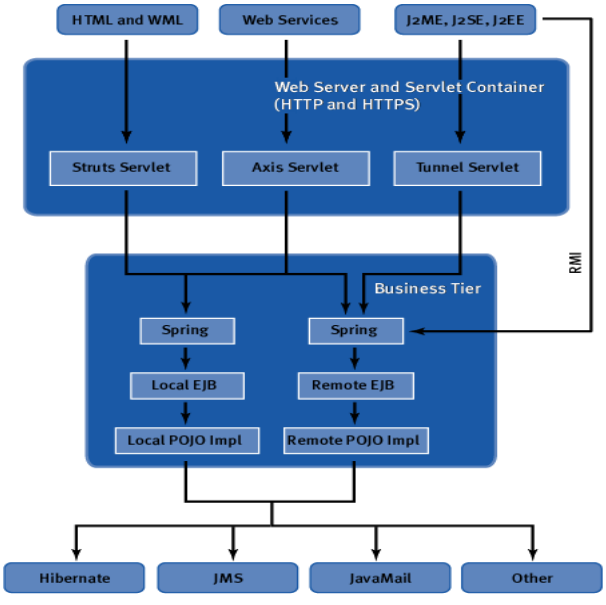

Figure 3: Architecture of Liferay, source [26].

Customisation of portal pages and potlets in some open sources Portal Framework e.g. eXo Platform [24] is not easy, and can involve a lot of configuration, but for Liferay layout management is very easy. Liferay Portal has a Web-based Graphical User Interface for user interaction to design the layout of Portal Pages without modifying any configuration files, which is similar to Stringbeans Portal.

Liferay Portal Enterprise comes with many useful portlets, and in fact Liferay portal has maximum utility portlets as compared to other open source Portal Frameworks, which are JSR 168 compliant and can be used in any Portal Framework with little changes.

Liferay supports WSRP specification as long as both WSRP consumer and WSRP producer are a Liferay portal instance. This is not the true essence of WSRP specification and WSRP support should not be only limited to Liferay. Liferay configuration requires some non-standard deployment descriptors which are mainly to accommodate Struts and Tiles, which can make development more complicated.

Like most of the other Portal Frameworks Liferay uses a default database Hypersonic 1.7, which is fine for development purposes but lacking in functionalities required for production use. Liferay can also be used with any database with minimum efforts due to the use of Hibernate [10] in its design. Liferay has JSP portal tag libraries and lot of utility classes in different packages to assist programmer in developing the portals/ portlets. Use of these utility packages eases the development of portals but then those portals are tightly coupled to Liferay and portlets are no longer JSR 168 compliant.

- **6.6 Stringbeans**

Stringbeans Portal [27] is composed of a JSR 168 compliant portlet container, and a framework for effectively administering portal applications. Stringbeans is deployed as a J2EE Web application in a container that supports Servlets 2.3 and Java Server Pages (JSP) 1.2 specification. For evaluation purposes Stringbeans was easily deployed and tested on Tomcat 5.0.28 servlet container by placing the Stringbeans Web Archive (WAR) file in the $TOMCAT_HOME/webapps directory, without any additional configuration. By default, Stringbeans uses a pre-configured Hypersonic database. Stringbeans should however work with any JDBC 2.0 compliant relational databases and was tested with PostgreSQL 7.4 database with satisfactory results, Stringbeans has no support for Hibernate [10], so shifting from one database to another database requires manual configuration. Stringbeans has a well documented set of user guides which can be browsed online or downloaded for later use, in fact Stringbeans has the best documentation of all open source portal containers tested. Online support from Stringbeans team was found very helpful and effective in terms of timely responses to bugs, queries and in implementing additional requested features. Stringbeans includes many user and developer friendly features and some of them are listed below:

- Easy layout management;
- Supporting themes for personalized look and feel;
- JAAS-based user authentication;
- Multi-column, menu, and full page layouts;
- Logging “user login” to simple file as well as a database;

- Per-portlet access control based on user ID, roles, and arbitrary database relations;
- Portal views based on user ID, roles, and relationships;
- Portlets capable of displaying RSS headlines, multi-page tabular data from database tables, reports, charts, XML documents via XSL transformations;
- Mobile client support (WML 1.1 and XHTML P1.0). However this has not been tested.

Stringbeans Portal can be deployed in a J2EE server with EJB container and we have tested Stringbeans Portal with JBoss 4.01sp1. Deployment was very easy and straightforward. Portlet deployment in Stringbeans Portal is very simple and truly JSR 168 compliant, which requires only two configuration files portlet.xml and web.xml. Most of the other Portal Frameworks come with many configuration files which make development and deployment very cumbersome, for example, JBoss Portal Framework requires as many as 6 to 7 different configuration files.

The current version of Stringbeans is 2.4.2 doesnot support WSRP specification, which means it cannot be used as WSRP producer and consumer; however there is an intention to provide a Web Services framework in future releases of StringBeans.

7. Evaluation Result

Following the evaluation criteria described in section 5, we have developed different portlets and hosted them in different Portal Framework, and these portlets are both test portlets and online live portlets available to users. Below is the result of our thorough and non biased evaluation:

Table1: Evaluation Result

Criteria	Portal Framework					
	Sakai 1.5	uPortal	Gridsphere	Exo	Liferay	Stringbeans
JSR-168 Compliance	0	5	5	5	5	5
Ease of Installation	3	5	5	5	5	5
Ease of Use	3	5	4	5	4	5
Documentation	2	2	4	3	3	5
Support Services	3	3	4	4	3	5
Administration of Portal	3	5	4	5	4	5
Customisation	4	3	4	3	5	4
Free Useful Portlets	4	3	4	3	5	3
Performance	2	4	3	4	3	3
Security	3	4	3	4	4	4
Technology Use	3	3	4	5	4	3
Portal Features	2	2	3	5	4	2
Server Dependency	3	3	3	4	5	3
WSRP Compliance	0	3	0	3	3	0
Total	35	49	51	57	58	51

8. WSRP Inter Framework Test

Different WSRP Inter Framework tests were applied in the uPortal and eXo Portal Framework to validate consumption of Remore Portlets. uPortal only provides WSRP consumer service, the basic uPortal WSRP test scenario was to deploy WSRP4J producer service as our test WSRP producer. WSRP4J uses reference implementation of Portal API called Pluto; therefore Pluto Portal Framework was working as WSRP producer by the mechanism of deploying proxy portlet. We deployed Information Service portlet and OGSA-DAI portlet which are Globus Toolkit (GT3) [28] based as remote portlets in the WSRP4J producer. The test results were very encouraging uPortal consumer can call the WSRP4J producer and get back the portlet mark-up from WSRP4J producer and display them properly in the browser.

To test the compliance of uPortal consumer with WSRP specifications, we deployed eXo Platform as WSRP producer and called different portlets from uPortal consumer. Results were satisfactory, although there were occasions when we failed to get the mark-up from eXo Platform producer.

When we did the test, we found that although uPortal and eXo framework claims to support WSRP, but there support is not 100% compliant to WSRP specification. It was found that if the portlet mark-up includes links referencing to external resource, these two Portal Frameworks will transform remote link to the local link. Another issue is the current WSRP mechanism does not properly supports user interaction like button actions.

9. Summary

We have outlined the work done in evaluating portals and portlets as a presentation layer to SOAs. The importance of the JSR 168 specification has been stressed as it makes it possible to deploy Java portlets in different Java Portal Frameworks without modification of the portlet source code.

In the second part we presented an evaluation of various open source Portal Frameworks, e.g. eXo platform, uPortal, Liferay, GridSphere and Stringbeans. Our comparison criteria are based on the ease of installation, user interface, standard of documentation, customisation and compliance to JSR 168 specification. We are now developing services and portlets to support the UK National Grid Service (NGS) and the JISC-funded Sakai VRE Demonstrator and have tested portability between all the frameworks mentioned. A separate evaluation of high-level frameworks for the development of VREs was recently completed for JISC and compared Sakai, CHEF, OGCE and GridSphere.

In the third part we outlined the results of a pilot investigation of WSRP based on inter-framework tests with WSRP4J, uPortal and eXo platform.

We will conclude the paper with a summary of experiences and observations and suggestions for further work. This will be explained in the context of the development and

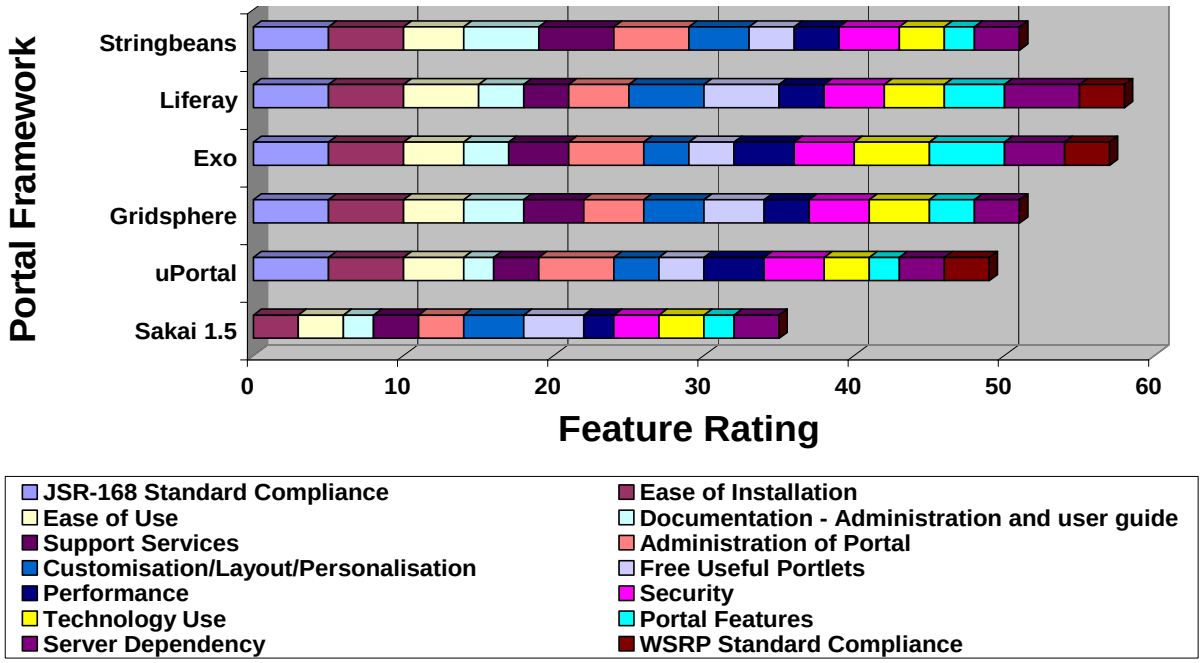

Figure 3: Evaluation Result as Bar Chart

deployment of JSR 168 compliant portals/ portlets for the NGS, VREs [29], ReDRESS [30], etc. Suggestions are made for the practical usability of WSRP which may form the basis of future in-depth investigations.

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.J. Allan, A. Hardisty, S. Wilson and A. Powell Service Component Classification, http://www.grids.ac.uk/ETF/public/WebServices/classes.html.

[3] Portal specification JSR 168 and API, http://www.jcp.org/en/jsr/detail?id=168.

[4] WSRP Specification 1.0 by OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp.

[5] Java Authentication and Authorisation Service, http://java.sun.com/products/jaas/.

[6] Java Open Single Sign-On, http://www.josso.org/.

[7] Struts, http://struts.apache.org/.

[8] Java Server Faces, http://www.jcp.org/en/jsr/detail?id=127.

[9] Spring, http://www.springframework.org/.

[10] Hibernate, http://www.hibernate.org/.

[11] Tiles, http://www.lifl.fr/~dumoulin/tiles/.

[12] Enterprise Java Bean, http://java.sun.com/products/ejb/.

[13] Web Services, http://www.w3.org/2002/ws/.

[14] Content Management System, http://www.jcp.org/en/jsr/detail?id=170.

[15] A. Akram, D. Chohan, X.D. Wang, D. Meredith and R. Allan, CCLRC Portal Infrustructure to Support Research Facilities, *GGF14*, Chicago, USA, 2005.

[16] R.Crouchley, A.Fish, R.J. Allan and D. Chohan *Sakai Evaluation Exercise.* Report to JISC (December 2004), http://www.grids.ac.uk/Sakai/sakai_doc.pdf.

[17] Sakai, http://www.sakaiproject.org/.

[18] Sakai VRE Demonstrator, http://www.grids.ac.uk/Sakai.

[19] uPortal, http://www.uportal.org/.

[20] Pluto, http://portals.apache.org/pluto/index.html.

[21] Central Authentication Service, http://www.yale.edu/tp/auth/.

[22] GridSphere, http://www.gridsphere.org/.

[23] eXo Platfrm, http://www.exoplatform.com/.

[24] X. Yang, D. Chohan, X.D. Wang and R. Allan, A Web Portal for the National Grid Service, *UK e-Science AHM2005*, Nottingham, UK, 2005, accepted.

[25] Liferay Portal Enterprise, http://www.liferary.com/.

[26] Java Messaging Service, http://java.sun.com/products/jms/.

[27] StringBeans Portal, http://www.nabh.com/projects/sbportal.

[28] Globus Toolkit, http://www.globus.org/toolkit/.

[29] M. Baker, H. Ong, R.J. Allan and X.D. Wang, Virtual Research in the UK: Advanced Portal Services.*UK e-Science AHM2004*, Nottingham, UK, 2004.

[30] 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.