

# Application & Tool Component Frameworks

Bill Olivier, June 2004

## Introduction

The JISC 3/04 Call seeks to fund user-facing tools that broadly support the areas: authoring of resources for learning; lifelong learning planning and ePortfolio systems; associated negotiation of learning; project-based collaborative learning; assignments and assessments; and personal learning environments that compliment and work with institutional learning environments.

Under the section ‘Technical framework for integration’, the Call suggests the need for ‘application component deployment frameworks’ (ACDFs).

The intent of such component frameworks is to:

- enable component functionality to be assembled according to need
- increase the flexibility and adaptability of user-level applications
- provide a top integration layer to the service oriented architectures of the eLearning (and eLibrary and eResearch) Framework Programmes
- allow process support to be more easily tuned, adapted or changed
- increase the reusability of funded developments across a wider community
- allow smaller, more focused projects
- enable projects to assemble and build on components produced by others

There are two broad approaches to such top level ‘application layer’ frameworks:

1. The provision of portals that support plug-in functionality on the server side, to be accessed through web browsers. They also provide a framework for integrating backend services.
2. Desktop application frameworks that that are extensible via plug-in components, and able to call on online services

This briefing elaborates further on these.

## Sustainability of Funded Projects

In the past, numerous innovative projects have been funded which have generally suffered from not spreading beyond their originators. One common factor is that they have been developed in a particular technical context: for a particular platform, in a particular technical environment with a unique mix of other systems and platforms. Such applications don’t travel well, as too much work and application-specific knowledge is needed to adapt them to work anywhere else.

To increase the spread of uptake and the sustainability of the outcomes of funded Programmes, it is proposed to work towards establishing common frameworks that will enable applications and services, from different sources, to work together and add up to more than the sum of the parts.

A major part of such Frameworks is developing, agreeing and standardising the technical interfaces that enable the various parts of the whole to be assembled and configured according to institutional needs and priorities, and to work together.

## **Distributed eLearning**

Not only is learning becoming more distributed within an institution, there is a significant increase in learning provision that crosses institutions and is provided on a regional or subject specialist basis. Examples include foundation degree courses, often provided jointly by HE and FE institutions on a regional basis, and modular cross-institutional post-graduate courses (e.g. in microprocessor design) where individual modules provide commercial short courses, but taken together can build credits for a post-graduate degree.

Provision of these forms of distributed eLearning, requiring shared information and resources for course development, learner information, enrolment, online support, assessment, and results present special difficulties requiring innovative solutions.

## **Open Desktop Tools and Applications**

Much of the development focus for learning, libraries and research has been on client/server applications. However there is recognition that most people still spend most of their time working with desktop applications. While the client/server model is good for finding, retrieving and sharing information, its not so good for authoring purposes which is generally done through desktop tools and applications. Even email, an archetypal network service, is best carried out using a desktop agent. Email is of course also made available using web mail, but usually this is used only under the necessity of travel or when no other alternative is available. Desktop tools are generally easier to use, more responsive and, with local storage, allow the user to continue to work in disconnected mode.

However, such tools and applications increasingly need to be able to communicate with other systems. In response, two trends are observable:

1. authoring is often a collaborative activity and thus support for collaborative authoring is increasing.
2. desktop applications are increasingly being enabled to use online services, such as publishing and data access to information, as well as more specialised services.

Several aspects of learning, outlined in the JISC 3/04 Distributed eLearning Call, would benefit from tools enhanced with these features. In particular they would form part of a Personal Learning Environment (see next). These can also be seen as providing the top 'user agent/applications layer' of the eLearning Framework, as set out in the documents of the parallel eLearning Framework programme (see more below). Such systems are intended to extend, enhance and compliment existing systems, rather than replace them.

## **Personal Learning Environment**

The notion of Personal Learning Environment is relatively new, but is emerging in response to some of the limitations of current first generation LMS/VLEs.

A PLE recognises that while Web-server based systems meet the needs of institutions, they do not serve so well the interests of lifelong learners. There are a number of reasons for this, which a PLE can potentially remedy:

1. Being browser-based, there is no record left on the learners side. It will be maintained on the institutional server, but lifelong learners will pass through

many institutions, perhaps more than one at the same time. Lifelong learners are the prime manager of their learning and have to co-manage their learning with each learning provider. There is therefore a strong case for learners needing their own system for planning, recording, reviewing and negotiating their learning path. The facility to enable a lifelong learner to select and transfer this information to others will depend on the use of Profiles such as the UK Transcript, the European Diploma Supplement, the Draft UKLeaP and IMS ePortfolio specifications all of which map to IMS LIP.

2. Browser-based learning requires a continuous connection with a server. However, fulltime and part-time, local and distance learners need, in varying degrees, to be able to continue learning when disconnected. For this they need a personal learning environment that enables them to download "learning chunks", work off-line with them, then re-connect and synchronise with the institutional learning environment. This will require a small service specifications w....
3. When learners move across institutions they are confronted with having to learn different ways of doing essentially the same things. A fully elaborated personal learning environment could provide the learner with a uniform environment that travels with them and plugs into an institutional environment via well defined service interfaces. This will require specifications for transferring conversations as well as content, but any VLE specific features would have to be used in online mode only.
4. Work created by learners, learning resources, emails and discussions are fragmented across different systems and applications. A PLE could provide a coherent means of pulling together and organising these disparate resources. In a fully developed form, it would integrate the learning activity and recording processes, archiving the structure content and products of the learning activities automatically creating a rich ePortfolio.
5. LMS/VLEs already face problems when scaling up the numbers of users. This is set to get significantly worse as new forms of learning, enabled by specifications such as SCORM 2004 and IMS Learning Design, come on stream, and further in the future, by dynamic personalisation of learning and intelligent tutoring systems. Borrowing from the thinking of the P2P world which uses the significant 'edge' capacity of users systems, and building on a PLEs capacity for disconnected learning, can be used to share the load with the server, dynamically offloading the personal parts of the learning process and running them on the users systems, communicating back outcomes.

At its simplest a PLE is just a collection of useful tools that are available to learners, but as the above suggest, these need to be integrated to create a coherent system that supports all aspects of learning, providing learners with a system that is fully complementary to and integrates seamlessly with institutional learning environments.

Two characteristics are required for PLEs:

1. They must be standards-enabled in order to communicate and exchange information with institutional learning environments.
2. In order to grow, evolve and be adaptable to different institutional environments, they ideally need to include a plug-in framework so that diverse forms of functionality can be provided and the learner's environment configured according to current needs.

While a number of eLearning specifications and standards are in place, many that are required for the full JISC eLearning Framework are still to be developed. Prototyping and elaborating these is a significant part of the eLearning Frameworks Programme.

Frameworks capable of supporting a PLE and other desktop applications for learners and teachers are still at an early stage and there is only the beginnings of any standardisation effort (see later). In the context of this programme, such frameworks are the subject of exploratory studies and further development which should inform future work and explore the possibility of convergence on a coherent framework, or if necessary frameworks. Applications that use existing frameworks may be developed as proof of concept, together with assessments of the chosen frameworks suitability for the task.

### **Relation to the JISC Service Oriented Frameworks Programmes**

Another important factor to be taken into account in the development of tools and applications for learners and teachers, is the use of services.

The JISC is supporting and coordinating Service Oriented Frameworks Programmes in the areas of eLearning (The eLearning Framework, drawing on IMS Web Services), eLibraries (The JISC Information Environment augmented by Z39.50, SRW and other service interfaces) and eScience (Virtual Research Environment building on the GRID/OGSA).

These are converging on the adoption of the Web Services model, and the intent is to build on this in order to integrate them, at least at the common services level. However, they may well diverge as they approach the application specific level in their respective domains.

There may also be apparently common services, such as repositories, that prove to need different interfaces for different application areas. For example the metadata, search and data transfer requirements for large research data sets and for learning objects may prove sufficiently different as to require different repository service interfaces.

But as far as possible, common approaches, common toolkits and common services will be adopted, so that institutions do not have to support arbitrarily different systems in the different domains.

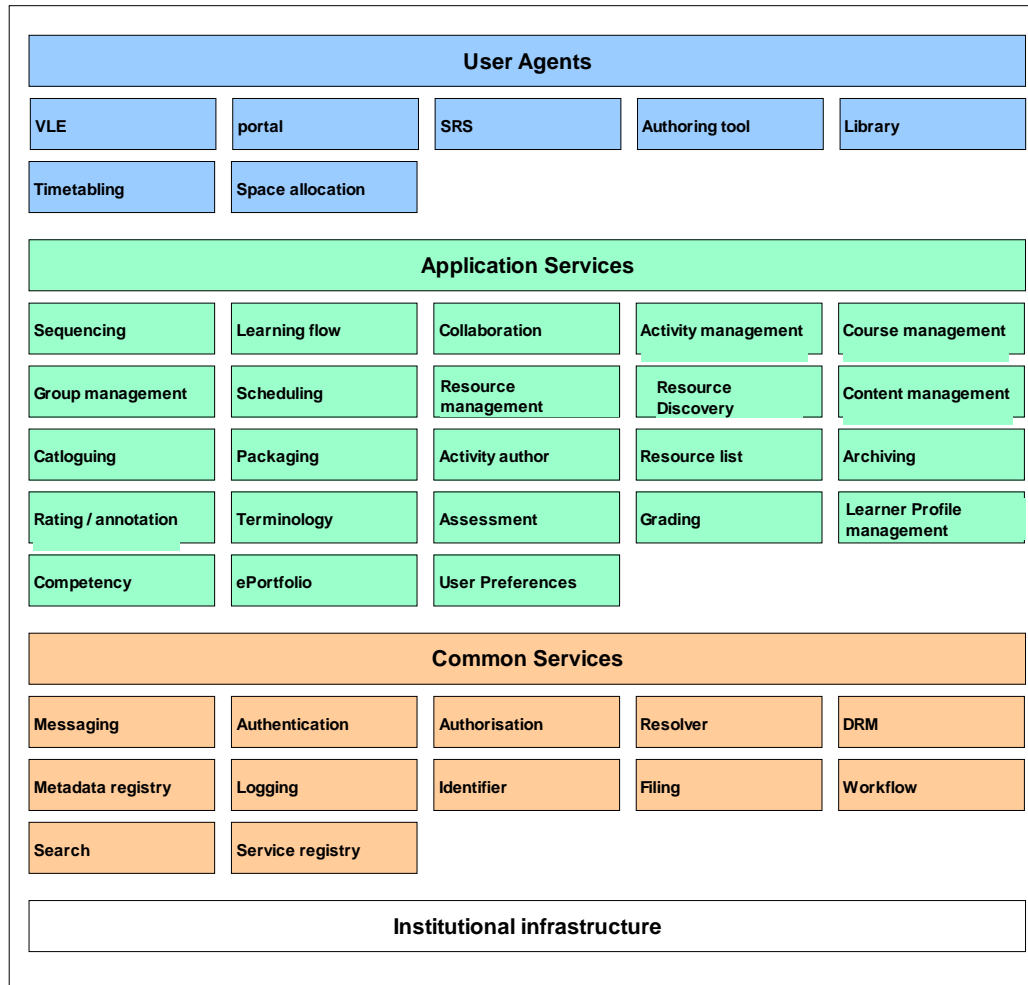
An important characteristic of these frameworks is that they are not fixed but evolving and will continue to evolve in the light of the outcomes of funded projects and other developments in the field. Projects are encouraged to think in terms of, and explore the use of frameworks. The programme is open with respect to these, and this document is a guide to these but is neither exhaustive or exclusive – informative rather than normative.

### **The Link between Applications and Services**

These frameworks all seek to define services that can be built on and used by user applications, as well as by other services. However user tools and applications are separated from of these services so that the service functionality need only be implemented once and then called on as needed, rather than be repeatedly implemented in every application. It is expected that early work will focus on exposing the functionality of existing institutional systems as Web-service interfaces that allow them

to be reused in multiple and more flexible ways. This works well within an institutional environment, but may not work so well for nomadic users or ‘ad hoc’ groups, such as are formed at conferences and other meetings, where all members are not part of the same cross-institutional trust framework (see later).

The following diagram is taken from The JISC Frameworks Whitepaper (URL). For the most recent version select ‘The Layered Services Framework’ from the Table of Contents on: <http://www.cetis.ac.uk/members/frameworks/index.html>



**Model of services demonstrating common and application services together with selected user agents (Figure 7)**

While the eLearning Frameworks Programme concentrates its efforts on the green and orange Services Layers, this programme can be seen as developing the blue ‘User Agent’ Layer. The examples in this diagram are a small illustrative subset of the wide range of applications that could be developed in this top layer.

Their common characteristic is that they call on the eLearning Application Services and Common Services in the two layers below.

A model that some are proposing, as the next layer over a service-oriented architecture, is one of workflows linking lightweight applications where the workflows can be relatively easily composed by non-programmers, using graphical UI to wire component

activities together, probably as a hybrid of document and activity style workflows. The type of application envisaged for this support semi-structured and repeating patterns of activities which can be set up quickly and then refined over time or adapted to meet changing needs. Applications designed to support specific institutional and learning processes benefit from user involvement in their development from an early stage, particularly when delivered in small incremental iterations that allow feedback and change of direction as needs evolve and become better articulated in the light of experience with early releases.

The link between an application and a service is a critical aspect of a service oriented architecture, and the toolkits being developed to support this are a key part of the strategy (see next).

The basic ideas for this application layer are well set out in a short book (DW 03-1), produced in a collaboration between SAP and O'Reilly. While it sets forth a new software integration component between the underlying services and the applications (the 'Enterprise Services Platform'), rather than applications talking directly to web-enabled services, much of the thinking applies to how a service oriented architecture can be used by applications. Another short companion book ((DW 03-2) spells out a rationale for the underlying service oriented architecture and would be appropriate for those have to make decisions on whether to go down this route. While broader in focus than just SAP, they are none-the-less strongly influenced by it and these books are therefore particularly interesting coming from a large-scale ERP system vendor as an indicator of their change in thinking and future direction.

## ***Web Services, Toolkits and APIs***

### **SOAP & WSDL**

Web Services generally use SOAP as the basis for the on-the-wire protocol. This enables a service written in one language to be accessed and used by another service or application written in another as SOAP is language and platform independent and there is now a general consensus building up that this is the way to support cross platform communications.

In addition SOAP interfaces for Web services can be defined using another standard, WSDL (Web Services Definition Language). One strong feature of defining an interface in WSDL is that there are code generators for various platforms that allow 'adapters' or 'proxies', which handle all the translation between the language and the SOAP XML, to be automatically generated. Such generators are available for both Java and .NET and generators for other languages can be expected to follow.

### **Toolkits and APIs**

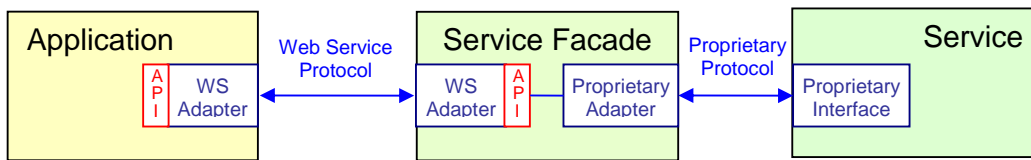
Such generators allow Web Service toolkits, with adapters for both the client and server sides, to be created relatively easily, but it puts a greater burden on defining the WSDL definitions well enough for the automated generators to be able to use them.

As well as a 'network-facing' Web service interface, these adapters also have, as appropriate, a client or server facing side. This takes the form of a language-specific API. Programmers have to add the adapter to their application or service and add the code needed to communicate with it by using this API.



Web Service and Web Service Client Toolkits

If it is not possible to directly integrate the toolkit adapter into the server software, it may be integrated into a Web service facade that translates to and from the Web Service protocol into the server's proprietary interface. As well as saving developers from having to write and test the adapter code, part of the intent is that loading an adapter should be user configurable so that alternatives and upgrades can be provided without needing a new release of the software.



Web Service Toolkit used to create a Service Facade

Clearly the Web Service interface and the programmer's API must both implement the same abstract data model and behaviour model. What is more this must be true across ALL toolkits for the same service.

IMS, which is currently developing a Web service definition for the Enterprise specifications, is using UML to define the abstract model and using WSDL for the bindings. CETIS, as part of its contribution to this specification, is developing an open source Enterprise Toolkits, generated from the Enterprise specification's WSDL bindings using the Java generators. Brockenhurst College is producing a .Net version of an Enterprise Toolkit as part of the first round of the JISC toolkit projects. These will be provided as open source toolkits with licenses that allow both open source and commercial use without royalties. It is intended that this toolkit should act as a prototype for further toolkits for other Web service based specifications in future.

The JISC Frameworks Programme will be developing further Web service interfaces, toolkits and services. The first early results of this programme should be available by October and bids in this Programme are encouraged to look closely at the outputs of these projects to see if they will be providing Web service interfaces that they can build on.

[http://www.jisc.ac.uk/index.cfm?name=elf\\_projects](http://www.jisc.ac.uk/index.cfm?name=elf_projects)

Defining APIs as part of a standard can also help both in the transition stages to Web Services and where, for various reasons, Web services may not provide a good solution. The APIs insulate tools and applications from the wire protocol and its implementation. For example a defined search and retrieve API might be used for a client adapter to Z39.50 repositories as well as for an adapter to repositories that support the newer SRW (Search/Retrieve Web Service). The Z39.50 adapter could be used with an existing repository and replaced, without impacting the rest of the client software, when it is upgraded to SRW. If both client and server use plug-ins they can both be up upgraded at the same time when an upgraded toolkit is made available (e.g. when security is added).

## ***Possible Implementation Frameworks***

It must be emphasised again that the Frameworks discussed are not a finalised given from the outset, but rather something that is evolving and being developed through the course of the next three years. This present a challenge for projects that are at the top ‘applications’ level for the framework. One approach will be for such projects to provide a Web services interface over existing systems so that their data can be exposed to end user tools and applications. Where this is not possible the proprietary interface should be accessed through a plug-in client adapter with a publicly defined API (see later).

### **Criteria**

A number of criteria are set out to guide the choice of application framework that can be used across UK F/HE.

1. *Cross Platform Portability.* A number of different desktop operating system platforms are used in UK F/HE, the major ones being Windows, Macintosh, and Linux. An application framework should ideally work across all of these platforms. That tends to mean that frameworks should be developed in Java, Python or other languages that use a portable virtual machine, or compiled languages with independent libraries that allow applications to be targeted at multiple platforms.

It can be argued that this is less important when it comes to portals that are accessed by browsers as the Web format provides cross-platform presentation and a human interface to any background services. However server-side solutions that are operating system and platform specific can only serve a subset of the FE/HE community

Conversely, platform independence is more important when it comes to frameworks for desktop tools and applications.

2. *Open & eLearning Standards Aware.* For eLearning tools and applications to work together in a distributed context, they need to make full use of eLearning standards. For more generic functions, open standards should be used. These standards can either be of the data variety, typically based on XML, or they can be behavioural and service oriented standards. Such standards can either be supported directly by the tools and applications or they could be supported in the framework.

3. *Extensible Frameworks and Plug-in Platforms.* Frameworks should at least be extensible by third parties and provide libraries and mechanisms for components to exchange data. Ideally they will also provide well defined interfaces that allow third party components to be plugged in and integrated into the environment.

4. *Built in Functionality and Libraries.* Frameworks should have appropriate functionality built in which can be reused by extensions, components, tools and applications. The more capable and appropriate the function set with respect to the domain or typical application, the more useful the framework is to developers.

5. *Open Source.* While this is not a hard requirement, in practice the few systems that meet these criteria are in fact open source.

5. *The Other –ities.* And the ideal platform also exhibits reliability, scalability, modularity, adaptability, etc.

## Portals, Portlets and Web-based Application Servers

This strand follows the Web-browser-as-universal-access model. All the action takes place behind the Web server. Such services are created using powerful backend development environments and have characteristics that meet many of the criteria outlined above.

### Portlet Standards

Portals are commonly used to provide application integration at the user presentation level, rather than at the system to system data level. Portlets are components that generate a part of a web page presentation for integration into the final web page that is presented to the user. Portal providers have developed different approaches to developing plug-in portlets, which, while making them extensible, create a new set of problems when trying to move them across different portal platforms. Two recent standardisation efforts seek to address these.

1. JSR 168 (Java)

This provides a set of standards for portlets, defining specifications for the way a portlet should plug into a portlet container and a set of portlet APIs that address personalisation, presentation, and security. An information oriented portlet might well gather information from a data source, transform it into a web page fragment for passing to a user and manage the user's session when they engage with the portlet.

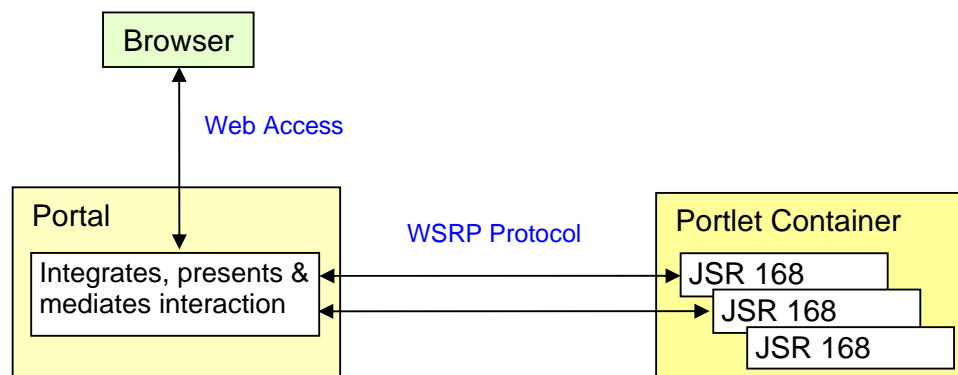
<http://www.jcp.org/en/jsr/detail?id=168>

2. WSRP (Web Service for Remote Portlets)

This differs from JSR 168 in that it specifies how a portal engine should communicate with a remote portlet provider using a Web service protocol as the means of communication between them. passes a portlet screen fragment to a portal engine.

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrp](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp)

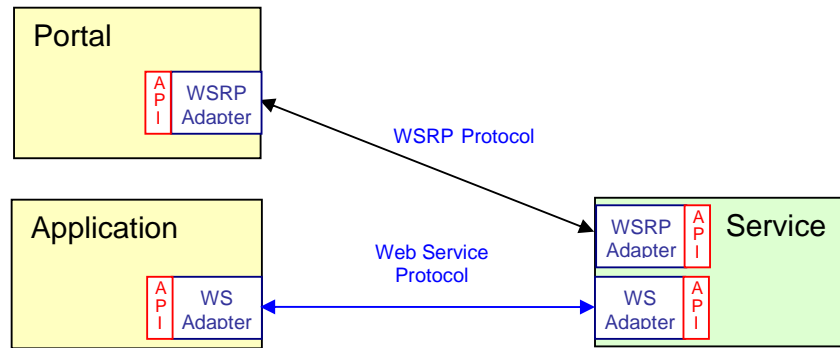
The roles of the two specifications are therefore different, and can be used together. The WSRP provides ready-made portlets to front end portal providers, who therefore have very little work to do. On the other hand they have no control over what appears on screen and how the user interacts with the remote service. The work of creating the portlet is carried out on the remote system and here, it would be possible for the provider to use JSR 168 to manage the portlet components that create the portlets.



WSRP and JSR 168 working together

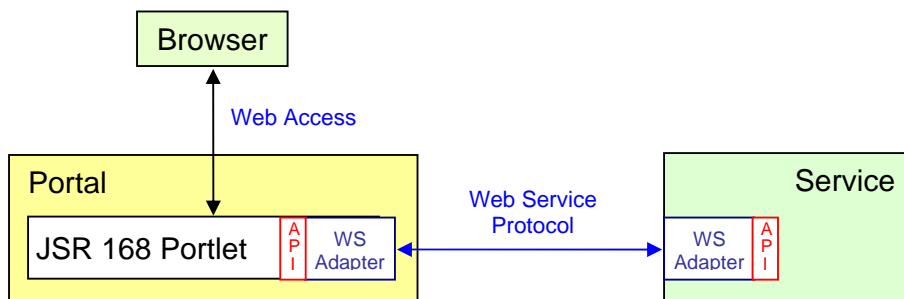
Thus WSRP can enable Java portlets to be used by non-Java Portals. A useful overview presentation covering this and other uses is available from the OASIS page for WSRP at <http://www.oasis-open.org/committees/download.php/3488/wsrp-overview-rev2.ppt> This still leaves open the question of how portlets should communicate with data sources.

In the case of WSRP, the Web service protocols are simply being used to effect communications between the portlet provider and the portal engine. There is no inherent connection between WSRP and Web services as a means for enabling a service oriented architecture and system-to-system communication and integration. A Web service typically provides access to data or other services for remote services or applications to use. It may or may not also provide direct WSRP access, but this would be one way of surfacing a Web service at the human level.



Service with both Web Service and WSRP interfaces

Another way would be for a portlet to link to one or more data sources via a Web service interface and this would fit with a Web service implementation of a service oriented architecture.



Portlet providing access to a Web Service

When taking a portal-based approach it is strongly recommended that JSR 168 and/or WSRP are used for portlets as these standards will allow exchange of portlets between systems.

There is a SourceForge project, POST (Portlet Open Source Trading) set up to enable the exchange of both JSR 168 and WSRP open source portlets.

<http://sourceforge.net/projects/portlet-opensrc/>

There are a number of portal systems that can be used with these standards and the number is growing. But rather than the portal engine used, the key factor is the

portability of the component portlets produced that will enable them to be reused across a number of different portal platforms.

## **The Sakai Project**

The Sakai Project is a \$6.8M community source software development project founded by The University of Michigan, Indiana University, MIT, Stanford, the uPortal Consortium, and the Open Knowledge Initiative (OKI) with the support of the Andrew W. Mellon Foundation. The project is producing open source Collaboration and Learning Environment (CLE) software with the first release in July 2004. The Sakai Educational Partners' Program extends this community source project to other academic institutions around the world.

The significance of the Sakai project in this context is that they are making a strong commitment to the use of the JSR 168 Portlet standard in uPortal from U Indiana <http://www.uportal.org/> as one part of a 'Technology Portability Profile' which also includes the OKI OSIDs (Open Service Definitions) <http://web.mit.edu/oki/>. The OKI OSIDs provide a set of APIs for adapters to a range of education related and common services which will provide a means for accessing backend services; a Tool Interaction Framework, based on U Michigan's CHEF collaboration framework server <http://chefproject.org/portal> which will allow components and tools to communicate with each other within CHEF; and a facility to customise Web based UIs independently of the content.

The OKI (Open Knowledge Initiative) played an important role in promoting the importance of service-based architectures and the role of APIs in the world of eLearning standards. The particular definitions they produced provide a starting point for further service based definitions. It is to be hoped that, as Web service based eLearning specifications are produced, the OKI OSIDs will be both taken into account and in turn kept aligned with the specifications as they are produced. They would then provide the APIs for the proposed toolkits and service adapters developed to support the open Web Service specifications and thus enable Sakai products to use these Web service oriented specifications in general, and, more particularly, to operate within the JISC Frameworks programme and conversely, enable Web services developed within the JISC programmes to work with Sakai.

CHEF is built on the JetSpeed portal engine, together with a number of other open source technologies. It too will be supporting the JSR 168 standard. CHEF meets many of the criteria set out above, but a key question is whether it will support open eLearning standards.

Sakai <http://www.sakaiproject.org/>

## **WSRPs used directly by Desktop Applications**

An interesting use for WSRP suggested in a set of Overview slides on the WSRP web site is that the WSRP Consumer can be built directly into a desktop tool or application, such as Word in the example given in slide 7.

<http://www.oasis-open.org/committees/download.php/3488/wsrp-overview-rev2.ppt>

This technique bridges the two portal and desktops approaches suggested here.

## ***Desktop Tool and Application Frameworks***

This section is an exploration of some possible frameworks that meet the criteria outlined above. Projects may suggest others.

### **Eclipse and NetBeans**

The first two frameworks, Eclipse and NetBeans, are more often thought of as Java IDEs (Integrated Development Environments). However both of them are designed as generic plug-in frameworks and in both cases the Java IDEs are supplied as plug-ins to the underlying platform. Both are open source, written in Java and hence cross platform.

They were both intended to support the more complex teams that are now needed to develop Web-based applications. With multiple authors with different skills using different tools, which have different ways of managing files, then integration and coordination can be a problem. It is therefore quite possible to unplug the JAVA IDE components and plug in other development tools in their place.

It was therefore part of the intent of both to provide a common platform that allows different tools to be plugged in so that the output of these different tools can be brought together using a coherent filing system. The filing system is itself a plug in which, as well as the local file system, can thus support FTP, CVS (Concurrent Version System) or WebDAV. In both cases, CVS is the default.

### **Comparison**

However there are differences between the two. The most significant difference is that, while NetBeans uses the standard Java Swing framework for its GUI, Eclipse has provided its own graphical framework, SWT (Standard Widgets Toolkit).

For those who remember the early days of Java, the AWT provided a thin skin over each platform's native widgets. This caused a number of cross-platform compatibility problems and Swing was developed as platform independent solution with its own widgets created from common platform primitives. However this approach came at the price of reduced performance.

Eclipse it seems has reverted to the approach of using native GUI widgets, but this time seem to have overcome the compatibility problems. The result is a livelier system and this is in part the reason for more programmers now appearing to prefer the Eclipse Java IDE over the NetBeans based one – Sun's Forte, now Sun ONE Studio, with Eclipse claiming some 6 million plus downloads to NetBeans 2-3 million.

It also means that it is harder to port tools already developed using Swing to Eclipse, although it can resent Swing UIs, but with a number of restrictions.

<http://eclipse.org/>

<http://www.netbeans.org/>

### **JSR 198**

A large number of add on tools are now available for both platforms. Some of these are commercial offerings which is permitted under both licenses. However most developers of tools would like to be able to add them to both platforms, and to other development platforms based on Java. Oracle therefore proposed a common interface which is now nearing completion under the Java Community under the title JSR 198.

This is due to be released over the summer of 2004. If it succeeds, it will a standard for plug-in desktop environments, in much the same way that JSR 168 is providing this for portlet plug-ins.

<http://www.jcp.org/en/jsr/detail?id=198>

## **Mozilla**

Mozilla, the name given when Netscape Communicator became an open source development project, has now progressed beyond providing just a browser and email client, into a powerful development platform for any kind of web-based applications – the “Collaborative Web”, as originally conceived of by Tim Berners-Lee, comes to mind. Although written in C/C++, Mozilla have gone to great lengths to make it cross-platform. Different compilations are needed fro each platform.

However a great deal of development can be done using the Mozilla platform without having to go into C programming.

There are many components to the Platform, but three significant ones are XUL, XPCOM and the use of RDF.

XUL (XML User-interface Language) provides a means of specifying all the UI widgets in the Mozilla platform using a declarative XML language and those who have used it claim it is a very efficient way of creating GUIs.

XPCOM stands for cross-platform components, of which there is a large number. These components can be accessed via scripting, but if there is not one that meets your needs, then new component can be added.

RDF (W3C Resource Description Framework) is used to handle all information aspects of Mozilla that takes place behind the GUI and these information structures are manipulated through calls into appropriate XPCOM components.

<http://www.mozilla.org/>

For in depth coverage of using Mozilla as a RAD tool, see:  
Rapid Application Development with Mozilla, Nigel McFarlane, Prentice Hall, 2004

## **Chandler**

Chandler is the main product of the Open Source Applications Foundation (OSAF), set up by Mitch Kapor of Lotus fame. Chandler is described as “a Personal Information Manager (PIM) intended for use in everyday information and communication tasks, such as composing and reading email, managing an appointment calendar and keeping a contact list.” It grew from dissatisfaction with currently available PIMs, and a recognition that such a system should be relatively easily adapted to meet different types of user.

OSAF took the decision to develop in Python, a fully object oriented interpreted scripting language (used for Zope, FLE3, etc.). It is designed to be a fully modular, extensible framework with well defined APIs.

[http://www.osafoundation.org/Chandler-Product\\_Roadmap.htm](http://www.osafoundation.org/Chandler-Product_Roadmap.htm)

## **HE extensions**

Its main interest in this context is that the Mellon Foundation funded a study into the potential of Chandler to meet the needs of (US) Higher Ed. This proved positive and provided a road plan for required the additions to Chandler. Mellon, together with the Common Solutions Group, a group of 20+ US universities that have pooled funding to develop applications that address their unmet needs, put up \$2.75 million to realise these proposals. This project is called Westwood within the OSAF.

See:

[http://www.osafoundation.org/Chandler\\_in\\_higher\\_ed\\_TOC\\_3002\\_05\\_13.htm](http://www.osafoundation.org/Chandler_in_higher_ed_TOC_3002_05_13.htm)

<http://wiki.osafoundation.org/twiki/pub/Chandler/WestwoodDesign/MellonGrantProposalAppendix.v.2.pdf>

Overall, Chandler looks like it will provide a flexible but powerful platform to meet more demanding needs for information management and sharing, that will both work in a P2P fashion as well as have built into the features required by HE and other educational institutions, such as security (Kerberos/Shibboleth) and nomadic users accessing through different machines, as well as mobile computer users with intermittent access.

## **Jabber**

In Chandler, it is proposed to use the Jabber protocol both for its more obvious functions of instant messaging and chat, but also as an XML-based channel for sharing information between Chandler systems. Jabber.org focus on the protocol and providing the open source Jabber server. The Jabber protocol is entirely in XML directly over sockets. They have made a version available that works over XML-RPC but have not gone as far as making it available over SOAP or Web services (which would probably be too heavyweight for live interactive exchanges).

<http://www.jabber.org>

## **XMPP**

Jabber has also provided the basis for the XMPP (eXtensible Messaging and Presence Protocol) Internet Draft standard recently approved by the IETF which tightens up the specifications and adds things like SASL. This has the potential to do for real-time messaging what SMTP did for email. Prior to the widespread adoption of internet email, it was only possible to email those who subscribed to the same provider. SMTP provided a way for the different email servers to communicate. We are currently at a similar stage with groupware and instant messaging, but XMPP should provide a route for connecting those servers together. The underlying structure of Jabber is based on email: individuals are provided with a unique address on a server. The servers act as peers and are able to forward (or store for later forwarding) messages to recipients on other servers.

<http://www.ietf.org/internet-drafts/draft-ietf-xmpp-core-24.txt>

<http://www.ietf.org/internet-drafts/draft-ietf-xmpp-im-22>

## **P2P**

For collaborative purposes fat clients and the use of P2P seems to be gaining in popularity, for many of the same reasons that people prefer to use email clients rather than WebMail.

## **Groove**

The best known example of a P2P collaboration tool is Groove, developed by Ray Ozzie the main architect of Lotus Notes. This provides a platform that is extensible by third

parties but already includes a fairly wide range of modular functionalities which are user selectable within each group. Some of its functionality is currently very limiting, for example, its calendaring is by group, rather than by individual which means that users have to re-enter their calendar data for each group they belong to. However, its main drawback, in this context, is that it is Windows only. It is also fairly slow, and if used on a Mac or Linux system using Windows emulation, its performance becomes unacceptable. It is also not suitable for large groups or large numbers of groups. While Groove is an interesting example of an extensible desktop platform, and not excluded as a basis to build on, its effective restriction to the Windows platform means that it cannot be recommended.

<http://www.groove.net/>

## **Hybrid Integration of P2P and Institutional Systems**

Already followed in Chandler, there is a tendency to produce hybrids of P2P and institutional systems. Servers are introduced into a P2P network to provide services, such as a presence service or a store and forward message service as in jabber, or a metadata index to resources on peers, as in Napster. Conversely, in the client/server world, fat clients take advantage of the power and capacity of client systems to offload work from central servers, as discussed earlier under the Personal Learning Environment.

### **LionShare: another Hybrid platform**

LionShare, like Sakai and Chandler, is another Mellon funded project that is producing a peer-to-peer collaborative file-sharing platform, based on the Gnutella open source, Java Limewire server (server+client).

It is a consortium Project based at Penn State, but with Internet 2 at Brown University, Simon Fraser University and MIT as partners.

It follows on from a three-year research programme into the use, or lack of use, of a large institutional image repository. It was found that it only represented 20% of the images on campus, with the rest residing on faculty members' desktops. An important reason they didn't use the centralised repository was that they wanted to retain control over who had access to their materials, which they shared, inefficiently and with some difficulty, via email and other means.

LionShare is producing a system that will be a hybrid system in two ways:

1. It is introducing a PeerServer into the peer network to allow metadata and file persistence for small groups which can otherwise suffer when members go offline, temporarily removing their resources from the group.
2. They are adding federated search and retrieval across both institutional and cross-institutional repositories, though their Canadian partners, EduSource ECL

EduSource have already produced tools that enable federated searching across a range of platforms, including using SRW over the JISC RDN hubs and DEST in Australia.

But a key feature is that they are integrating security via Internet2/Shibboleth into both the P2P network and to the federated search.

This they feel will both address faculty concerns about the use of their materials and at the same time address the feature of P2P networks that gave them a bad name through

Napster, namely lack of security. LionShare is consciously taking anonymity away from P2P systems – you know who you are providing access to and who is taking your files.

The LionShare peer will therefore provide a platform enabling users to create groups and share files with other members of those groups. They plan to add better ways of organising and finding files than is currently possible with desktop filing systems.

They are also being encouraged by Mellon to look at integration with Chandler and they are similarly looking at Jabber for peer collaboration.

They plan to have a release 1.0 available in October, which would make it a possible platform for building on.

<http://lionshare.its.psu.edu/main/>

<http://www.edusource.ca>

## **Security Frameworks**

Essential for distributed working both within and across institutions.

LionShare/Shibboleth Model: Users authenticate and obtain a certificate. This is used to authenticate the user to others based on a trust network and access control. It is intended to enable secure collaboration within and across institutions

LionShare's implementation of Shibboleth is providing one of the major use cases for Shibboleth2

## **Exploring Integration**

As the LionShare peer is in Java and Chandler is being developed in Python, there is a question about how they can be integrated. One avenue to explore is Jython, a port of Python to the Java Virtual Machine. Essentially it is a Python interpreter that generates JVM byte code. However it is also more than this, in that it handles casting Python Data types to Java, and allows Java packages to be imported and Java APIs interfaces and objects to be invoked from Python. It also supports constructors, introspection, use of Java Beans, streams, Swing, events, JDBS, applets and most other aspects of Java to integrated into Python.

Scripting languages are good for integrating modules and tools to create applications for specific purposes and Python may come to play this role in Java. It was voted the favourite option by NetBeans community when asked what scripting should be built into NetBeans and Jython is now provided as part of the NetBeans scripting module.

This in turn suggest the possibility of integrating the Chandler type of email, IM and calendar type of collaboration tool into an editor, shared file repository type of collaboration tools to produce a highly effective collaboration environment.

Similarly there is a project, entitled Blackwood, within Mozilla to develop improved ways of integrating Mozilla into a Java environment.

## References

### Service Oriented Architectures and Applications

(DW 03-1) *Packaged Composite Applications*, Dan Woods, O'Reilly, 2003

(DW 03-2) *Enterprise Service Architecture*, Dan Woods, O'Reilly, 2003

Already mentioned above, these books are a good guide to the rationale behind the SOA/Web services approach. Particularly relevant to those who have to consider the strategic aspects of, the business rationale for, and the development, effective use and deployment of a Web Services oriented approach.

### Web Services

*Web Services: A Manager's Guide*, Anne Thomas manes, Addison Wesley, 2003

There are plenty of technical books on programming Web services but this is different.

This book provides a clear, well written, no hype guide to the evolving world of Web services, written by someone with in depth experience. Good for those preparing bids for JISC funding in this area!

Programmers coming new into Web Services would also benefit from this book before (or after) diving into the details.

### Eclipse

*Contributing to Eclipse: Principles, Patterns and Plug-Ins*, Eric Gamma and Kent Beck, Addison Wesley, 2004

Written by the Design Patterns man and the Xtreme Programming and Test Driven Development Man!

If you are going to add things to Eclipse this is first. Not many open source projects can produce books like this.

*Java Developer's Guide to Eclipse*, Sherry Shavor et al, Addison Wesley, 2003

Written by the Eclipse training team in IBM. Adds more detail. Read second.

### NetBeans

*NetBeans: The Definitive Guide*, Eric Tim Boudreau et al, O'Reilly, 2003

Up to O'Reilly's usual standard – and seems to be the only book available on NetBeans.

### P2P

*Peer-to-Peer*, ed: Andy Oram, O'Reilly, 2001

Almost a manifesto, with many of the key players contributing, it gives an overview coverage of the field, some projects and protocols, and discussion of a number of issues. No code anywhere.

*Java P2P Unleashed*, Robert Flenner et al, SAMS, 2003

One of those big multi-authored books, it is reasonably recent, and thus covers more recent developments, notably the use of Web services in P2P context. Although it is ostensibly for programmers and uses Java, the amount of Java is relatively small, especially in the first half with a lot of focus on architectures and issues which interested non-programmers may also find useful. However, its Java orientation means a only passing mention of some mainstream P2P protocols such as Gnutella, FreeNet and Jabber, while going into depth on other Java-related ones such as JXTA, Jini and JavaSpaces.

### Jython

*Python Programming with the Java Class Libraries*, Richard Hightower, Addison Wesley, 2003

Very much the hands on tutorial style. Good for anyone with Java programming wanting to learn Python or explore its integration with Java.

### Jabber/XMPP

*Programming Jabber*, D J Adams, O'Reilly, 2002

This book is not so much about implementing Jabber in a programming language, so much as how to use the XML protocol defined by Jabber. It there is useful whatever programming language is being used.. But predates the XMPP effort.