

Plug-In Platforms for Lightweight Applications that Integrate Web Services

Bill Olivier, June 2004 (revised Oct 2005 and Aug 2006)

Introduction

Effective use of web based services requires user-facing tools that are able to integrate multiple web services into lightweight applications that support a wide range of functions for Learning, Research, Libraries, Information Services, management and Administration. This in turn is facilitated by the use of component based, plug-in. platforms, which provide common functionality and UI support, as well as well defined extension mechanisms and interfaces.

The intent of such component platforms 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 increasingly being used in the e-Learning, e-Research, e-Library and Repository spaces
- allow support for processes 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, both open source and commercial

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

1. **Web Portals** that support **Portlets** which provide plug-in functionality on the server side, to be accessed through web browsers. They also provide a platform for integrating and using backend services.
2. **Rich Client Platforms** that are extensible via plug-in components and able to also provide a platform for integrating and using backend services.

This briefing elaborates further on these.

Sustainability of Funded Projects

In the past, numerous innovative projects have been funded which have 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 broader uptake and sustainability of the software outputs of funded Programmes, it is proposed to work towards establishing common user level software environments 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 environments 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 e-Learning and Distributed e-Research

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.

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

Similarly the Internet has greatly facilitated the ability of researchers in different institutions and different countries to collaborate. e-Research developments have also adopted the use of services to support computation and data management distributed across grids.

In addition, as Researchers are frequently also Teachers, there is a need for a user level tool and service integration environment that can be used for all these purposes.

Open Desktop Tools and Applications

Much of the development focus for learning, libraries and research has been on Web browser/Web server applications. However there is recognition that most people still spend most of their time working with desktop applications. While the Web browser/server model is good for finding, retrieving and sharing information, it is 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, more capable 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.

Relation to the JISC Service Oriented e-Framework

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 a service oriented e-Framework for Education and Research. This is an information framework that integrates the service specifications for e-Learning (The e-Learning Framework, drawing on IMS Web Services), e-Resources (The JISC Information Environment augmented by Z39.50, SRW and other

service interfaces) and e-Research (building on the GRID/OGSA/WSRF service specifications).

These are converging on the adoption of the Web Services model, and the intent is to build on this in order that these services can be integrated into a wide variety of applications as desired. To this end a consistent approach to services and their use is needed, and where essentially the same services are used across domains such as learning and research, then they should have the same interfaces rather than arbitrarily different ones.

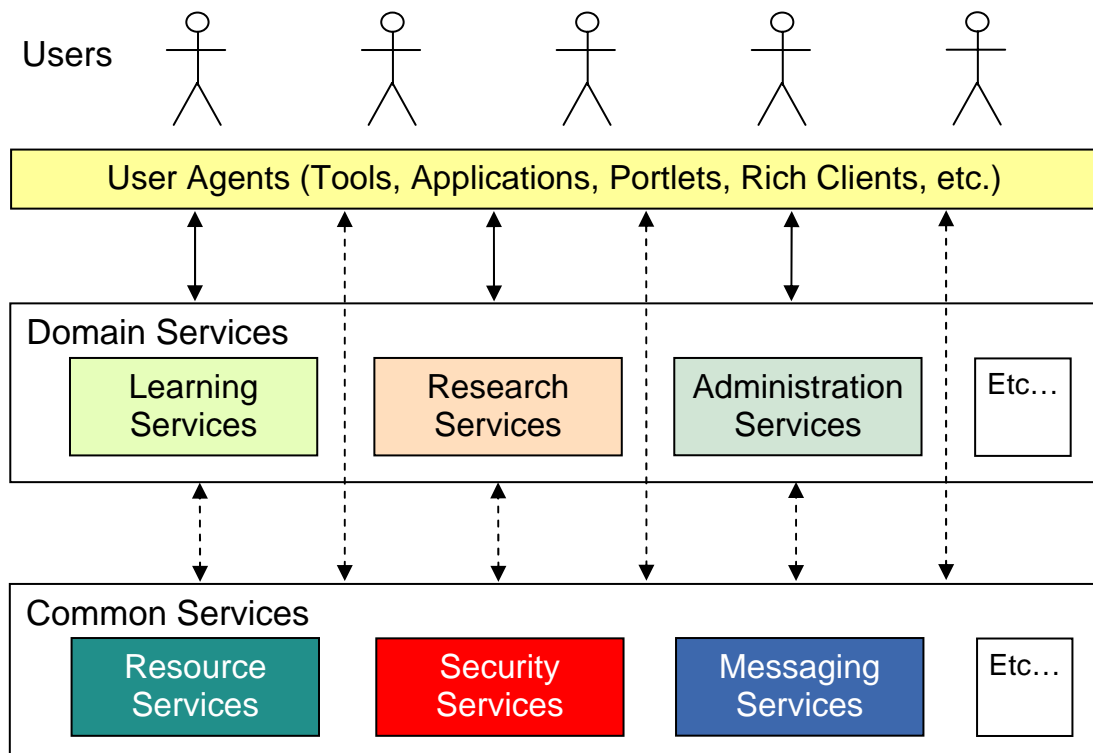
However, there may be services that are apparently common, such as repositories, where in certain cases, the requirements unique to a domain demand different interfaces for specific application areas. For example the metadata, search and data transfer requirements for large research data sets and for learning objects may prove sufficiently different as to require different repository service interfaces.

But as far as possible, common approaches, common Adapter 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 the e-Framework's knowledge base is that it is 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 contributing to, evaluating and exploring the use of the e-Framework.

The Link between Applications and Services

The developments in e-Learning, e-Research and e-Resource management all seek to define services that can be built on and used by user applications, as well as by other services. User tools and applications are separated from 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 collaborative groups, such as are formed at conferences and other meetings, where all members are not part of the same cross-institutional trust federation (see later).



Lightweight User Tools, Applications, Portlets, etc.
 call on domain specific services, such as Learning, Teaching or Research,
 and either through them, or directly, they call on Common Services

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 a graphical UI to wire component services together, probably as a hybrid of document and activity style workflows. The type of applications 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 Adapter 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, Adapter 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.

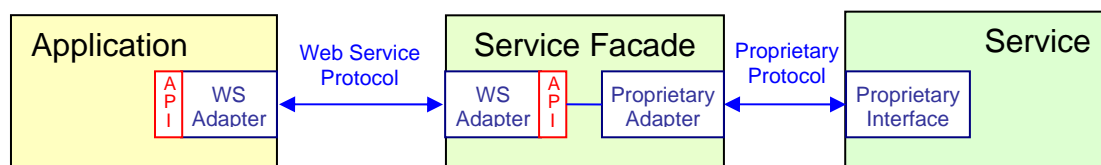
Adapter Toolkits and APIs

Such generators allow Web Service Adapter 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 Adapter 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 Adapter 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, in developing a Web service definition for its Enterprise specification, used UML to define the abstract model and using WSDL for the bindings. CETIS, as part of its contribution to this specification, developed an open source IMS Enterprise Adapter Toolkit, generated from the Enterprise specification's WSDL bindings using the Java generators. Brockenhurst College produced a .Net version of an Enterprise Adapter Toolkit as part of the first round of the JISC toolkit projects. These were provided as open source toolkits with licenses that allow both open source and commercial use without royalties. These toolkits acted as prototypes for further toolkits, developed under the JISC e-Learning Stream, for other Web service based specifications.

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 and other bindings may be preferable. The APIs insulate tools and applications from the wire protocol and its implementation. For example a defined search and retrieve API might be used to provide a programming language interface 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 the service is upgraded to SRW. Alternatively, where a client needs to make use of both Z39.50 and SRW repositories, it could include both adapters and switch as needed.

If client and server both use plug-in adapters, they can both be up upgraded at the same time when an upgraded toolkit is made available for example when security and / or reliable messaging is added to the standard for the service interface.

Possible Implementation Platforms

It must be emphasised again that the e-Framework discussed above is not a given that is finalised from the outset, but rather something that continues to evolve and develop over time. This presents a challenge for projects that are at the top ‘applications’ level of the e-Framework as the required services may not exist. One approach will be for such projects to provide a Web service interface to an existing system so that its 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 platform 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 platform should ideally work across all of these platforms. That tends to mean that platforms 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 platforms for desktop tools and applications.

2. *Open Standards Aware.* For tools and applications to work together in a distributed context, they need to make full use of 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 platform.

3. *Extensible Software Plug-in Platforms.* Plug-in platforms 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.* Platforms 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 platform 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.

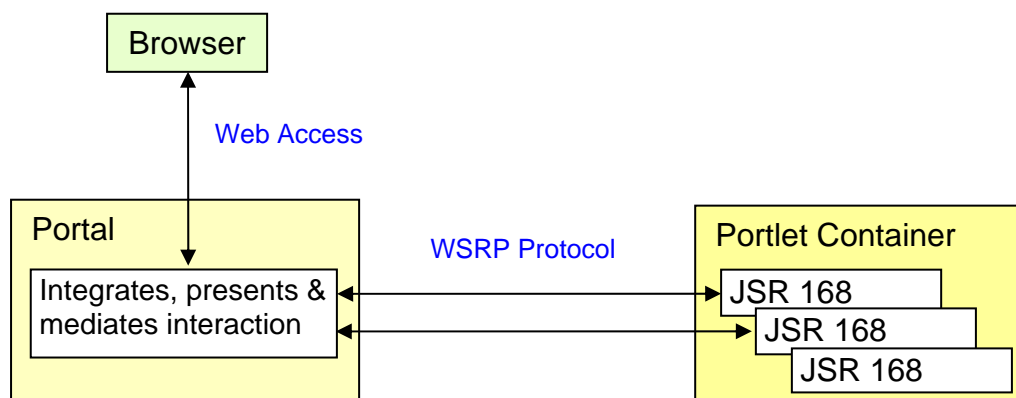
<http://www.jcp.org/en/jsr/detail?id=168>

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

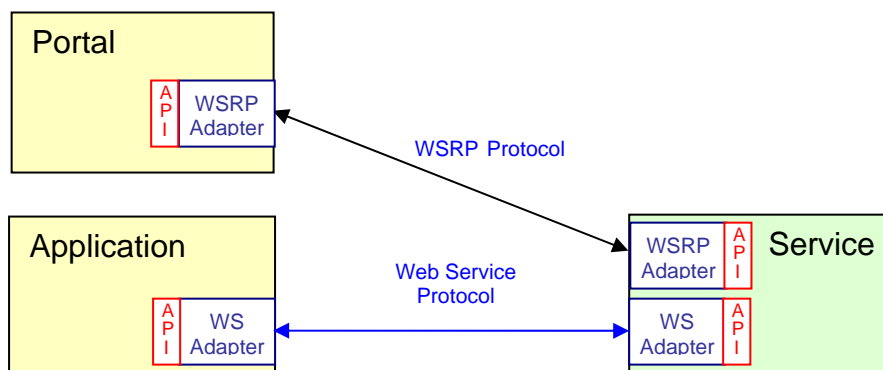
The roles of the two specifications are therefore different, and can be used together. The WSRP provides ready-made portlets to front end portal providers, who therefore have very little work to do. On the other hand they have no control over what appears on screen and how the user interacts with the 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

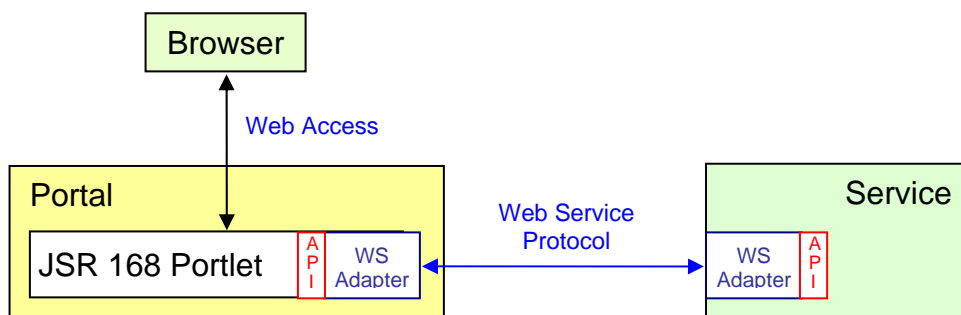
Thus WSRP can enable Java portlets to be used by non-Java Portals. A useful overview presentation covering this and other uses is available from the OASIS page for WSRP at <http://www.oasis-open.org/committees/download.php/3488/wsrp-overview-rev2.ppt> This still leaves open the question of how portlets 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 has produced and is continuing to develop an open source Collaboration and Learning Environment (CLE) software. 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 using the JSR 168 Portlet standard in uPortal from U Indiana <http://www.uportal.org/> as one part of a Technology Portability Profile (TPP) 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 e-Framework 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 would unify the use of portlets in both the portal and desktop approaches suggested here.

Desktop Tool and Application Platforms

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

Eclipse and NetBeans

The first two platforms, Eclipse and NetBeans, are more often thought of as Java IDEs (Integrated Development Environments). However both of them are designed as generic plug-in platforms 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 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 present Swing UIs, but with a number of restrictions.

One key advantage of Eclipse is its adoption of OSGI (Open Services Gateway Initiative) as a means of automatically downloading, installing and thereafter updating plug-ins. This effectively removes the objection to 'fat clients' (Rich Clients), namely the manual overhead of installing and updating applications on a large number of desktop systems.

<http://eclipse.org/>

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

JSR 198: A Standard Extension API for Integrated Development Environments

A large number of add on tools are now available for both platforms. Some of these are commercial offerings which is permissible 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 was finalised earlier this year under the Java Community under the title JSR 198.

The final version, released in May 2006, is worth exploring. If it succeeds, it will become 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 for each platform.

However a great deal of development can be done using the Mozilla platform without having to go into C programming, and these higher level programming components are cross platform. This has led to the suggestion that Mozilla, now available as separate email (Thunderbird) and browser (Firefox) applications, potentially provides a good rich client platform for Web service, particularly Web 2.0, based applications.

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 a 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 platform with well defined APIs.

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

Chandler is intended to provide a flexible but powerful platform to meet more demanding needs for information management and sharing; to 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 to support nomadic users with different machines, as well as mobile computer users with intermittent access.

While it may yet achieve these ambitious objectives, recent reports suggest that progress has been slower than originally expected.

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 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. 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.

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.

However the blocking tactics of the big Instant Messenger providers are succeeding in limiting progress.

<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 rich 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 formerly the main architect of Lotus Notes, and now CTO at Microsoft, following its absorption of Groove. Groove 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 the HE context, is that it is Windows only and is thus unable to support Mac, Linux and non-Microsoft mobile/PDA platforms. Early reports were that it was not suitable for large groups or large numbers of groups, but this may be addressed in newer releases.

The latest version of Groove is being released as part of MicroSoft Office 2007 suite, which will significantly increase its availability.

<http://www.microsoft.com/office/preview/programs/groove/highlights.msp>

While this has various extensibility mechanisms, it is not yet clear whether this will include facilities for integrating Web services. However projects may wish to explore its capabilities in this area, but will be limited to the Windows platform.

<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 re-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.

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 has produced a system that is a hybrid system in two ways:

1. It has introduced 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 have added federated search and retrieval across both institutional and cross-intitutional repositories, though their Canadian partners, EduSource ECL

EduSource had 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 was that they integrated security via Internet2/Shibboleth into both the P2P network and to the federated search. The participation of Steve Carmody of Brown University, one of the Shibboleth architects, meant that LionShare's implementation of Shibboleth was one of the main use cases for Shibboleth2.

They aimed both to 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.

LionShare Release 1.1 available in October, which would make it a possible platform for building on.

LionShare is also being used in the JISC funded SPIRE project which has integrated LionShare as a plug-in into the RELOAD Learning Design editor which in turn is built on the Eclipse Rich Client Platform.

<http://lionshare.its.psu.edu/main/>
<http://www.edusource.ca>
<http://spire.conted.ox.ac.uk/cgi-bin/trac.cgi>

Security Federations

These are essential for secure, distributed working both within and across institutions. LionShare integrates the 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.

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.

Understanding SOA with Web Services, Eric Newcomer and Greg Lomow, Addison Wesley, 2004

This does a good job of introducing and bringing together SOA, Web Services and business processes in an intelligible way. But it also manages to cover some more advanced topics such as reliability and transactions.

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 one is different. It 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

(For programmers)

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 the Eclipse environment, read this first. It provides the rationale and principles as well as the practice. Not many open source projects can produce books like this.

The Java Developer's Guide to Eclipse, 2nd Edition, Jim D'Anjou et al, Addison Wesley, 2004

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

Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications, Jeff McAffer, Addison Wesley, 2005

If you are going to use the Eclipse RCP as the basis for developing tools or applications, this is a good place to start.

NetBeans

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

Up to O'Reilly's usual standard, but getting dated.

NetBeans IDE Field Guide: Developing Desktop, Web, Enterprise, and Mobile Applications, 2nd Edition, Patrick Keegan, Prentice Hall, 2006

The second edition seems to be the only book covering JavaBeans v5.0.

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, and thus covers many 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 there is 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.

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 therefore is useful whatever programming language is being used.. But it predates the XMPP effort.