Sakai Architecture Evaluation

Adrian Fish
Lancaster University Centre for e-Science (`http://e-science.lancs.ac.uk`)

This document assesses the software comprising the Sakai virtual learning environment and is an outcome of the JISC funded Sakai Evaluation project. The assessment is based on RC1 and may not fully apply to RC2.

1 Introduction

Sakai (`http://www.sakaiproject.org`) is a software designed to add collaboration and course management facilities to the uPortal portal framework. The software already provides collaboration tools in the form of chat, discussion and shared file space, and is being extended with further tools designed to add course management functionality. Tools are being added for both test creation and assessment, the unspoken ambition being to create a functional open source competitor to Blackboard.

2 Look and Feel

Sakai takes the basic portal paradigm of portlets arranged in a grid formation on screen and adds the ability to group users into *worksites*, viewed as tabs across the top of the portal display. Sakai diverges somewhat from the concept of allowing a user to arrange several portlets on screen and then persisting that layout between user sessions. Instead, Sakai displays one portlet at a time and arranges a toolbar of buttons down the left hand side of the display. While on the surface the fact that you can no longer aggregate you own collection of portlets seems to be a waste of good portlet functionality, I think that having the portlets accessible quickly on a toolbar is a sound design decision.

3 Architecture

Sakai currently has hybrid architecture aimed at allowing the gradual transition from CHEF style tools to Sakai style TPP (**T**ool **P**ortability **P**rofile) tools. Whereas CHEF uses Jetspeed as its portlet layout engine, Sakai uses uPortal. Whereas CHEF used the Apache project's Turbine as its component and persistence framework (Model), the Sakai TPP uses Spring (although there are rumours of Apache's Avalon being used instead. Whereas CHEF uses Velocity as its display (View), the Sakai TPP uses JSF (**J**ava **S**erver **F**aces). Whereas CHEF uses Struts as its intermediary between the components and the display (Controller), the Sakai TPP uses JSP (**J**ava **S**erver **P**ages). The hybrid architecture arises from the requirement that CHEF tools also need to run alongside these TPP, so all the technologies mentioned in this section, bar Turbine, are present in the Sakai software stack. This hybrid approach enabled the basic CHEF collaboration tools to be brought straight across from CHEF with minimal modifications. This decision allowed the core Sakai team

to work on architecture, rather that tool porting in the initial stages of the project. The downside of this is that the Sakai stack is large and more complex than it would be with a single coherent MVC framework in place.

Sakai is currently in a state of flux, the chosen architecture for the software is being debated upon and there is a drive to prioritise the establishment of WSRP (**W**eb **S**ervices for **R**emote **P**ortlets) compliance above the establishment of JSR 168 (pluggable portlets) compliance. The Sakai core team are also working on uncoupling the software from particular portal frameworks and Java application servers (current uPortal and Tomcat respectively). While these developments will ultimately be good for Sakai, they

There is currently a dialogue occurring, amongst the members of the SEPP (**S**akai **E**ducational **P**artners **P**rogram) and the core development team, regarding the architecture of Sakai. This dialogue was in part triggered by cross language support concerns raised at the first SEPP conference in Denver. What is now being proposed is a de-emphasising of JSR 168 (pluggable Java portlet code)compliance, with a corresponding emphasis on WSRP (**W**eb **S**ervices for **R**emote **P**ortlets) support. The reason for this is obvious; WSRP is language agnostic, so if Sakai can aggregate remote software services via the WSRP mechanism, it no longer can be accused of being Java specific. Another big shift in architecture is a proposed effort to uncouple Sakai from uPortal as its layout engine. There is now talk of being able to run Sakai with other portal frameworks like Jetspeed and Liferay. All of these shifts in architecture could be looked upon with concern, but what it highlights is the open nature of the devlopment effort and the influence that the SEPP members wield as a group. Sakai will now work towards being more easily integrated into existing institutional infrastructures, with eased deployment and tool aggregration facilities, and that can only be a good thing. The downside to these changes is that early access tool developers like ourselves are faced with the frustration of a shifting architecture to code to.

4 Strategy

Sakai is currently the aggregation tool of choice for the ReDReSS project and the current instability is causing problems with tool development at the present time. What is proposed is a spreading of risk during the further development of the ReDReSS portal, using the following steps:

1. Continue to run a stable version of Sakai on the ReDReSS website. This will continue to supply collaboration facilities and a reasonably structured place to hold project material. The instructions for installing such a Sakai instance are contained in another document (sakai_installation_guide.pdf).
2. Install GridSphere (`http://www.gridsphere.org`) with the aim of using this installation to test the JSR 168 compliance of any tools the project produces.
3. Write any tools in such a fashion that both JSR 168 and Sakai TPP versions are generated at compile time. The TPP versions get tested in an isolated Sakai instance (Adrian Fish's workstation) before live deployment. The JSR 168 version gets deployed into GridSphere and thus tested for compliance.
4. When Sakai reaches the point of architectural stability, any tools deployed in GridSphere are moved to the latest Sakai version running on the ReDReSS web server.