SAKAI Technical Report

August 2004

Introduction

This report is as an objective view of the publicly available download of SAKAI Release Candidate 2.

The goals of the SAKAI project are to produce the following:
- Specification for a framework allowing plug & play components to be deployed
- Developer toolkit for new SAKAI friendly tools to run in the framework
- Reference download to demonstrate the concepts and for anyone to use under license.

The scope of this paper is not evaluate how the available code has met these objectives since the web site is clear that they will not be met until SAKAI version 2.

The current distribution is essentially an updated version of the CHEF tool being exposed through the uPortal content aggregation system. CHEF was developed at the University of Michigan as a collaborative tool for teaching.

Future releases will include as assessment system called Samigo and a content delivery mechanism replace OnCourse, and Open Courseware.

Architecture

The architecture assessment is made from the existing source code made available from the SAKAI project site and documentation from the SEPP meeting in July.

The Tools Portability Profile (TPP) is a specification for constructing end user facing components that can plug directly into any SAKAI compatible framework. The specification is a recommended use of the core technologies (see below) and a package structure for deploying components. The specification is not sufficiently detailed or stable to ensure true portability across implementations at this stage. This concern was recently highlighted in the re-prioritization of WSRP instead of JSR168 for portlet aggregation.

The biggest concern for the current code structure is the lack of a domain model. The current code and documentation for plugging in components is neutral of any domain or information model. As a result, the current release looks like yet another development framework.

The concerns over the domain model may be addressed with the introduction of the Samigo assessment model as the first education related component to be used in the SAKAI framework. It is hoped the design team demonstrate good practice in separating the domain model from the implementation dependencies.

The current distribution file system layout does not reflect the package naming structure and as a result is somewhat confusing. The purpose for the layout is not clear other than supporting a distributed development team where new components are simply created at the file root level. This approach makes it very difficult to see how a shared information model will be supported.

There is significant code duplication in the current distribution but that is most likely a result of the combination of two major projects in a short time scale and will be addressed in later releases.

The reference code looks to have been unit tested without the use of JUnit as a testing framework. As well as being best practice, the inclusion of a JUnit test harness structure would better support the development of components for use in the framework providing a consistent way to test for interoperability. This observation is made from the numerous places where a *main* function used for testing has been commented out.

Core Technologies

The core technologies of the framework are a collection of widely accepted best practices for java based web applications:

The component architecture is based on the Spring Framework (www.springframework.com) which is a widely recognized framework for developing web applications without a full Java 2 Enterprise Edition (J2EE) managed container. The downside is that the Spring Framework is only recommended over J2EE when the full systemic qualities and service levels of enterprise software are not needed.

The presentation layer utilizes Java Server Faces (JSF) for implementing the Model View Controller (MVC) design pattern and ensuring the presentation and user interaction is abstracted from any application business rules.

Configuration management is driven by the Apache Maven tool for build and deployment. The scripts are available with the reference download and are written for deployment into a Tomcat container with an in-memory XML database or Oracle setup.

Conclusions

The project is still in very early stages and the team is focused on delivering a working solution for the Universities of Michigan and Indiana. It could be argued that the SEPP initiative should have been delayed until the architecture and design was stabilized given the short-term focus on a deliverable system.

The problem is that these are really two separate projects attempting to share a single constrained resource and not necessarily having the same vision. Without a formal mechanism for collecting requirements from the SEPP members and allowing the architecture team time to refactor the design, the program will never realize it's vision of a truly open shared platform.