1 Another independent Technical Report

This report from an unknown source had been circulated to a few people and was obtained by us under an agreement that it would not be circulated further. Since a number of decisions may have been taken subjectively based on its content we however consider it defensible to publish the text here with our comments on the factual content in our objective evaluation of Sakai.

```
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.
```

Some comments on this report follow.

"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 TPP is designed to make tools portable across Sakai installations, not across other portal frameworks. It is specified in enough detail to enable this portability.
- The architectural switch to WSRP demonstrates the efficacy of the SEPP process. The switch is designed to permit the crafting of tools in other languages, and the exporting of Sakai worksites directly into WSRP consumers. For more information please refer to *sakai_architecture_evaluation.pdf*.

"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."

- Sakai is domain agnostic. Sakai is designed to provide a stable software framework into which to plug tool components, together with a useful set of collaboration tools pre-plugged. It is

NOT a development framework in the sense of JSF, Struts or other programming libraries, it is more of a production line framework - just plug your tools in and customize for your institution. The domain specific elements, and the required models, come with the tools.

"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."

- There is a significant amount of FUNCTIONALITY duplication in the Sakai codebase. This is because a pragmatic decision was taken to write adapters for the existing (and production tested) CHEF tools to allow them to work within the new framework, whilst coding up new, 'native', versions.

"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."

- The unit testing used in Sakai is a mixed bag. There are some JUnit tests in the legacy calendar tool code, and quite a lot of ad hoc tests within main functions. A point worth noting here is that unit testing is a well understood art and does not need JUnit to be successful.

"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 SEPP could well have been delayed. This would have given advantages, as well as disadvantages. It would have meant less distractions for the core team, but could have also cut out a major source of user feedback. The SEPP serves several purposes; it provides user input,testing and development funds. Ultimately the SEPP could be seen as a useful outcome of the Sakai project as it is hoped that it will result in new models for educational collaboration on open source sofwtare development. The mechanism for requirement gathering probably does need formalising to a degree, but the foundations are there. There is a dedicated team for eliciting user requirements and producing GUI wireframes for the software team; this is probably more than many software houses have.
- Sakai and CHEF are not two separate projects, what we are seeing is the gradual transformation of CHEF into Sakai, as stated on the Web site.
- A final point to make is that there are several effective players in the VLE market. If Sakai delivers on its promises and is marketed successfully, it will take considerable market share from other VLE companies like Sun, Blackboard and WebCT.