Technical Report
Sakai Project

Sakai Architecture

October 14, 2004
Version 1.0

Craig Counterman
Glenn Golden
Rachel Gollub
Mark Norton
Charles Severance
Lance Speelmon

www.sakaiproject.org

Sakai

Introduction

System architecture consists of the abstract concepts that constitute an application, environment, or proposed framework. Sakai is no different. In order to provide a better understanding of Sakai, this document defines an Abstract Sakai Architecture. Technology is specifically absent from this description, as that is what a design consists of. The Sakai Java Framework is an example of an implementation strategy that can be drawn from this.

The Sakai Architecture does not specify any technology - to see an example implementation of this Architecture please look at the Sakai Java Framework document.

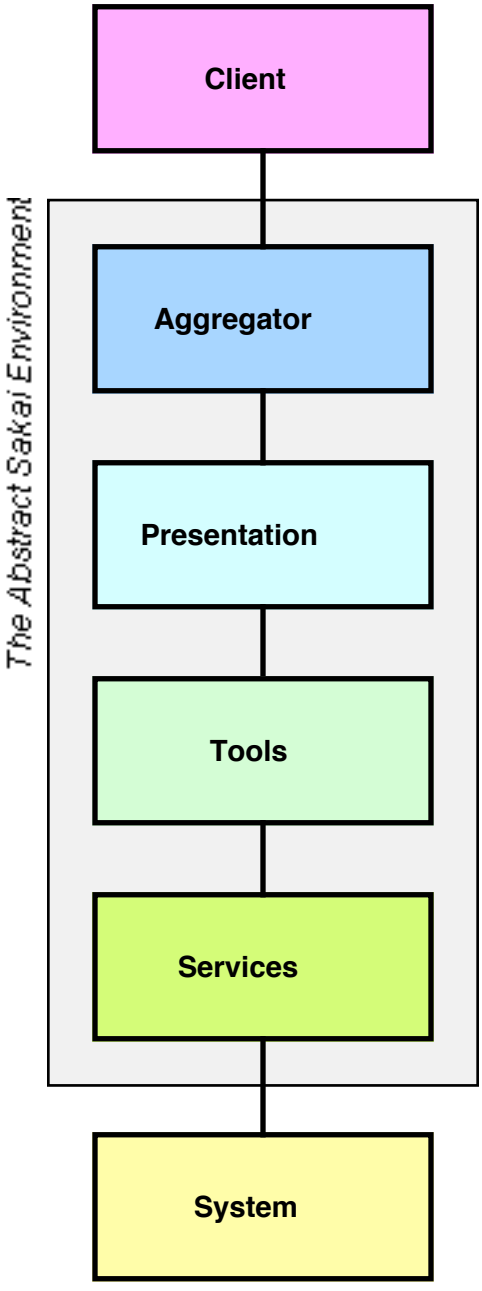
Principles

The architecture presented here is intended to provide design support for the following guiding principles:

> Create a system in which different kinds of applications, some potentially outside of the Sakai environment can be aggregated to create a single user experience.
>
> Provide for separation of application and presentation logic.
>
> Provide an environment that allows tools and services to be migrated and re-used between other Sakai environments, and potentially other (non-Sakai) environments.
>
> Capture educational, application, common, and system capabilities into re-usable services that can be migrated in Sakai and non-Sakai environments.
>
> Create an environment that allows tools and services to be adapted to local system requirements including enterprise and back office services.

Architecture Diagram

The Sakai architecture consist of the following elements:



Client

Sakai is intended to be run as a client / server application pair. While the majority of the clients will be standard, off the shelf web browsers, customized browsers and other

network aware applications may be used in some situations. The majority of Sakai applications will have their output aggregated and presented to the client using a markup language, such as HTML Specialized clients may communicate directly with Sakai services provided they are enabled for such transactions (content authoring, for example).

Aggregator

The output of one or more Sakai application (and potentially non-Sakai applications as well) can be combined using an aggregator server application. This aggregator allocates and manages screen real estate and certain user interface transactions (to be defined). Support for accessibility is provided using a combination of standard UI elements in the presentation layer and the aggregator.

Presentation

The presentation layer combines data from a Sakai tool and a user interface description to create a mark up fragment that is aggregated before delivery to the user. The user interface description is ideally contained in a resource external to the software and makes use of standard user interface elements designed to deliver a consistent Sakai user experience.

Tools

A Sakai tool is an application that marries presentation logic with application logic contained in services. Tools provide code to respond to user interface requests and events, which may (or not) modify data managed by services. The tool may draw on services to provide data to the presentation layer.

Services

A service is a collection of classes that manage data via a defined set of behaviors. This data may or may not be persistent across user sessions. Where possible data should be modeled and represented using accepted industry standards. Behavior is represented using a published Application Programming Interface (API). Services may call other services, creating dependencies on them. Services are intended to be modular, reusable and portable across Sakai environments, and potentially to non-Sakai environments also.

System

The system is the server environment that the Sakai environment resides, plus any remote capabilities available to it. This environment may include web servers, database servers, operating systems, file and resource repositories, enterprise and back office systems, etc.

Revision History

Version No.	Release Date	Comments
0.5	July 29, 20004	Initial Document - written by Mark Norton.
1.0	October 14, 2004	Modifications and formatting for the 1.0 release.