Sakai Software Organization

Please direct questions or comments about this document to:
Charles R. Severance, csev@umich.edu and Joseph Hardin, hardin@umich.edu

One of the major goals of the Sakai Foundation is to produce production-quality collaboration and learning software. This document describes how this activity is organized and provides some guiding principles around how the Sakai software is developed. This document only covers software projects - not the other activities within the Sakai Foundation.

Many of the patterns in this document (and even some of the text) are derived from Apache documents (http://www.apache.org/foundation/how-it-works.html). Readers of this document are encouraged to read and review the Apache procedures to have a better understanding of the underlying principles that govern this document.

In terms of software development approach, Apache and Sakai overlap in many basic ways. In the areas where Sakai and Apache overlap, the Apache ideas are adopted nearly verbatim. The primary place where Sakai is different than Apache is due to the fact that Sakai integrates a large number of its sub-projects into a single release. Few of the Sakai sub-projects are useful outside the context of the Sakai framework. Most are intended as components that are assembled together to produce the Sakai final product.

As a result of this additional need for a unified, production-quality release across projects, Sakai requires additional choreography and orchestration *across* projects that are unnecessary in Apache. Where this choreography is needed in Sakai the attempt is to add it in a way that is true to the proven Apache principles guiding projects.

Sakai Software Projects

Much like the Apache Foundation, Sakai's software effort is divided into projects. Each software project within Sakai is governed by a Project Management Committee (PMC), which is composed of the committers for the particular project. The PMC chair for a project is the lead committer for the project. The Sakai Foundation board delegates the technical decisions for each software project to the PMC for that project.

Projects in Sakai have different granularity - some are quite large and have broad impact (like framework) while others are relatively isolated and only affect a simple application within Sakai (like rwiki).

No single project is "above" or "below" any other project. All projects are peers within Sakai and when one project has a requirement that can only be satisfied by some change in another project, the two projects must work together to come up with mutually

agreeable solutions that meet the needs, timelines, and requirements that are mutually agreeable.

Apache reference: http://www.apache.org/foundation/how-it-works.html#structure

Phases of Sakai Software Development

Sakai software development operates in two distinct phases: (1) Normal Development and (2) Release Preparation. Because a Sakai release is an activity which crosses many software project boundaries and is a highly time critical activity, release preparation is a more centrally controlled operation than normal development.

The orchestration and timeline when Sakai switches from normal development to release preparation is handled through close cooperation between the Sakai Project Coordinator, Sakai Chief Architect, Sakai Release Manager, and Sakai QA Director. Timelines for releases are discussed and decided with the community and the Project Coordinator maintains and communicates the release schedule.

Normal Development

During a "normal development" phase, Sakai Software Projects operate with relatively loose coupling. The Chief Architect and Project Coordinator communicate across projects to help align work efforts and get those work efforts into shape for an appropriate upcoming release.

During normal development, the Release Manager and QA Director focus on producing maintenance releases for the prior releases by tracking bug fixes and including them in maintenance releases as appropriate.

In the current Sakai approach to releases every six months, we expect to be in normal development for 3-4 months.

Release Preparation

The signal to switch from normal development to release preparation is an activity called "Integration Week". Integration week comes 4-5 weeks before the intended release date. Integration week is a time where all Sakai Software Projects focus on getting their code ready for the feature freeze and QA period. During integration week developers from the most active projects often get together and work out the remaining small details of any cross-project dependencies. Also it is quite common for those projects which are "ahead" when integration week starts, to spend time helping the projects which are "behind," so that all projects finish properly by feature freeze.

Feature freeze is at the end of the integration week and results in the first release candidate. QA begins with the release of the first release candidate and lasts until the release is completed. During the QA phase, the Release Manager is fully responsible and

empowered to make tactical decisions about the release. This includes setting priorities on outstanding issues (such as blockers) and making calls as to whether a particular software project is ready for the release.

Once the release is completed, generally there is a few weeks where the Release Manager and QA Director continue to track high-priority bugs and work with each Software Project to get fixes for those bugs into the first maintenance release.

Generally, a few weeks after the release is completed, the project goes back into Normal Development mode under the guidance of the Project Coordinator and Chief Architect.

Roles In the Software Development Process

One of the major differences between Apache and Sakai is the need to produce a single release of Sakai, integrating across projects. It is important to realize that the two phases described above are often done by different organizations in many open source projects. In Sakai we do both activities in the same organization. In a way, Sakai is both the Linux project and Red Hat in the same organization.

This leads to several important Sakai Foundation roles in the Sakai project that have no direct analogues in the Apache project.

Sakai Chief Architect

The primary purpose of the Sakai Foundation Chief Architect is to insure the long-term technical coherence of the ultimate released Sakai software product(s). The Chief Architect works with the framework project and many other projects to insure that Sakai continues to "fit together" in the long term.

The Chief Architect publishes guiding documents and technical roadmaps to guide Sakai developers. Roadmaps are not precise project plans, but instead a way to ensure that cross-project technical directions are well-communicated to all projects to best allow each project to be able to react to cross-project technical issues in a timely manner.

While the Chief Architect has broad responsibilities to communicate and interact across projects, she has no particular direct authority over any of the projects. Like any other member of the community, the Chief Architect must work cooperatively with the PMC for each project.

The Chief Architect may be called in to help resolve disputes between projects, but in this role, she is only offering her guidance and acting as a facilitator to help the conversation between the projects move forward.

Since the Sakai Framework project has significant impact across the Sakai release, typically the Chief Architect will work closely with the lead committer in the Framework project.

The Chief Architect is not automatically granted committer status in any Sakai project. Like any committer, the Chief Architect must earn their committer status by their contribution and commitment to the particular project. The Chief Architect also has no particular say in the makeup of the commit list in any project.

The Chief Architect is responsible for producing a report each quarter that summarizes her activity and provides a forward-looking architecture roadmap. The roadmap is expected to be at a high level and lets the community know the likely directions in the next 6-24 months in terms of architecture. This report is targeted at the developer community and published broadly.

There is no analogue in the Apache project for the Chief Architect role. The Chief Architect role is one of orchestration across projects. By focusing the Chief Architect role on communication and not granting any particular authority to the Chief Architect the role is designed to enhance and support the Apache-style PMC structure used in Sakai.

Sakai Project Coordinator

The primary goal of the Sakai Foundation Project Coordinator is to track community activity across projects and help different activities work best with each other and work best with the community. The PC is expected to be a single point of contact about "who is doing what" in the Sakai community.

The PC actively tracks and communicates community and foundation activity through all phases of software development. The PC is lightly involved in nearly every aspect of Sakai.

The project Coordinator is responsible for a regular report (preferably online and continuously updated) that all members of the community can review.

Sakai Release Manager

The primary purpose of the Sakai Foundation Release Manager is to produce a high quality release of Sakai in a timely manner. The Release Manager has very specific responsibilities when Sakai is preparing a release. Once integration week begins and the release process is underway, pretty much the Release Manager (and his release team) is in charge of the process and will make decisions as necessary to produce a quality release on time.

The release team looks at outstanding bugs and decides which bugs are important to be fixes for the release. The team works with the projects to get these bugs fixed in a timely manner. During this period the Release Manager is really making "demands" on project teams during the release process. If these demands are not met, then the respective project's software contribution will probably not be included in the current release. This is about as close as Sakai gets to "issuing orders".

However even in the release process, aspects are loosely coupled. The release manager cannot override the Software Project Committer team. The only recourse that the Release Manager has if the PMC does not respond to the Release Manager's request is to either drop that project from the release, include a previous version of that project, or include the product in the release in conditional form.

Once the release is completed, the project switches back to normal development mode and the Release Manager focuses on the maintenance branches for the release. The Release Manager continues to work with the projects identifying bugs, getting the fixes to the bugs, seeing that those fixes are aggregated into a maintenance release, and then properly tested and released.

The Release Manager will produce a report for each major release summarizing what was done for the release, any major issues with the release, and any outstanding problems identified during the release that we would need to address in the future.

QA Director

The primary responsibility of the Sakai Foundation QA Director is to insure the overall quality of the Sakai product over time. The QA Director manages and coordinates the volunteer QA efforts for each of the Sakai releases.

The QA Director also assesses Sakai's overall systems and approaches to delivering a quality product, makes changes to the QA process that she can, and provides recommendations to the community as to how processes can be improved which will result in overall quality being improved.

The QA director will produce a report for each major release summarizing the major QA activities for the release and identifying any outstanding issues that will need addressing going forward.

Roles Within Project

The Chief Architect, Project Coordinator, Release Manager, and QA director are the cross-project roles and as such there are no equivalent roles to be found in the Apache Foundation. Within a Sakai project, however, the structure and terminology is identical to the Apache Foundation.

Apache reference: http://www.apache.org/foundation/how-it-works.html#roles

User

From Apache: A User is someone that uses our software. They contribute to the Sakai projects by providing feedback to developers in the form of bug reports and feature

suggestions. Users participate in the Sakai community by helping other users on mailing lists and user support forums. The passive users are also known as lurkers.

Developer

From Apache: A Developer is a user who contributes to a project in the form of code or documentation. They take extra steps to participate in a project, are active on the developer mailing list, participate in discussions, provide patches, documentation, suggestions, and criticism. Developers are also known as contributors.

Committer

From Apache: A Committer is a developer that was given write access to the code repository and has a signed Contributor License Agreement (CLA) on file. Not needing to depend on other people for the patches, they are actually making short-term decisions for the project. The PMC can (even tacitly) agree and approve it into permanency, or they can reject it. Remember that the PMC makes the decisions, not the individual people.

PMC Member

From Apache: PMC member is a developer or a committer that was elected to the PMC due to merit for the evolution of the project and demonstration of commitment. They have write access to the code repository, the right to vote for the project-related decisions and the right to propose an active user for committership. The PMC as a whole is the entity that controls the project, nobody else.

Note: The primary reason to have a difference between the Committer and PMC member is to point out the fact that simply given write access to a Sakai Software Project does not bestow on that person the right to make final decisions for the code. People will get commit for many reasons - one example is someone who works across many projects on Configuration management, Accessibility, UI cleanup, or Internationalization. That person may technically have the ability to make changes everywhere in the code. However that person does not have the right to make significant design decisions or rewrite major portions of code without working with the PMC.

The PMC is best though of as the "senior committers" for a Sakai Software Project and hold complete responsibility for their project. If a committer who is not part of the PMC makes a change, the PMC may veto that change and back it out.

Many projects will dispense with any differentiation between committer and PMC members and simply define the committers as the PMC. For a small project with 3-4 committers, there is no reason to get too wound up in process and procedure. The distinction is still useful to reinforce the notion that the decision making within a Software Project is delegated to those who have demonstrated long-term involvement and are clearly dedicated to the project in the long term.

Becoming a Committer

The initial committer list and PMC is generally comprised of the people who developed the software in the first place. Once the initial committer list is in place, the committer lists and PMC maintain their own membership. Neither the Sakai Foundation Board nor the Sakai Staff have any influence in the commit list of a particular project.
The most general path to becoming a committer and PMC member in a project is that a person is a valuable addition to the team. In many projects there is a great deal of work to be done and so new people who are willing to commit to the project and help are generally welcomed with open arms.

One of the key aspects of the Sakai (and Apache) committer process is that it is intended to insure that the new committers will be a positive and somewhat long-term addition to the team. The process has a natural timeline as an individual moves through the user, developer, committer, and PMC member phases. Part of the purpose of this time factor is to insure that the person is indeed committed to the project for the long run. If a person simply needs to make a few changes in a project and then go back to come other project, then there is no need to grant committer status or include the user in the project's PMC.

The PMC makes its rules for membership, but this might be an example rule-of thumb: A PMC applicant should be willing to invest at least 1/3 of their time for 12 months on the particular software project in question. A PMC applicant should be willing to work on all aspects of the Software project including fixing bugs, applying other developer's patches, testing, and participating in the release process for the project.

Each PMC must establish and publish the rules for applying for membership. Looking at Apache PMC charters, there are three common patterns that Apache PMC's use: (1) Anyone can self-nominate themselves to the PMC and the PMC votes with a +1,0,-1 style, (2) only PMC members can nominate a new committer - then the PMC votes with a +1,0,-1 style, or (3) PMC membership is invitation only - the PMC monitors users and developers working on the project and contacts potential new committers as the PMC sees fit. There is no one right way to maintain the list - these examples serve to emphasize the point that the PMC gets to determine the rules of its membership.

Being a Committer

Apache references: http://www.apache.org/dev/committers.html

What must I do first?

The very first thing you need to do is to complete and submit a Contributor License Agreement (CLA). The process for this is described in a separate document. (See …)

What are the responsibilities of a Committer?

From Apache: As a Sakai volunteer, you have the right to set your own priorities and do the work that scratches your own itch. As a Committer, you have a responsibility to the community to help create a product that will outlive the interest of any particular volunteer (including yourself). This means, for example, that the code that you commit should be clear enough that others not involved in its current development will be able to maintain and extend it. It also means that you are responsible for helping to grow and maintain the health of the Sakai community.

Deciding on release plans and releases
A prime responsibility of the Committers is to decide when a branch of code is ready for release. A release is not to taken lightly; each release must uphold the Sakai tradition of quality. Each Project Management Committee formally authorizes the distribution of releases to the public.
Applying patches
In order to grow and maintain healthy communities, committers need to discuss, review and apply patches submitted by volunteers. The Committers are also responsible for the quality and IP clearance of the code that goes into SakaiFoundation repositories.
Helping users
Committers should monitor both the dev and user lists for the projects that they work on and (collectively) provide prompt and useful responses to questions from users.
Monitoring commits and issues
Committers should review commit email messages for their projects and point out anything that looks funny or that may bring in IP issues. Monitoring Jira for bugs or enhancement requests is also a responsibility of Committers.

Note: this is an incomplete list and not authoritative.

Is there a set term for acting as a Committer? Will I have to be elected again?

From Apache: Merit never expires. If you become inactive for a time (usually six months or more) your account may be deactivated for security reasons. Most projects allow reactivation of committer status by application to the PMC.

Some projects use the concept of an emeritus committer status. This is typically suitable for those committers who can no longer give the time they feel is required.

New Projects

Sakai encourages new projects, as this is the source of the innovation necessary to produce the rich product desired by the Sakai community. The Sakai "contrib" area is Sakai's version of SourceForge. There are four basic phases in a new project's lifecycle. Not all projects will progress through all four phases: (1) completely independent contrib. project, (2) community adoption, (3) provisional status, and (4) a full component of the Sakai release.

Independent Contributed Project
This is a project that is using Sakai's SVN contrib. to support a Sakai-related development activity. It is relatively simple to get a new contrib. space created - contacting the Project Coordinator with a short summary of who will be the single point of contact (the initial lead committer) and what activity will be done in the space is all that is needed.

During this initial phase, there is no need for contributor agreements nor precise licensing requirements. But if this project is intended for ultimate release as part of Sakai the effort would be well-served to start out requiring Sakai Contributor Agreements and develop the code following all of the Sakai Foundation Licensing Practices. It is generally easier to start with this discipline at the beginning rather than trying to get this sorted out as the project is trying to move into Provisional or Release status.

The three common uses of contrib. are to: (1) develop a tool or component that will grow and mature for the ultimate Sakai release, (2) store useful information in a place that the community can look and reuse - a common example here is for a university to store their Sakai providers in a contrib. area because there is community interest in their local work, and (3) advance some completely crazy experimental Sakai work that a few people want to keep in one place so that they can work collectively on the effort and possibly recruit new members of the community to join them in their quest.

The Sakai Project Coordinator monitors and reports on these activities. If after a time, a contrib project seems to have no value and/or no activity, the effort may be archived and mothballed.

Community Adoption
Once a project reaches a maturity and quality level that the team feels shows it is ready to release, the tem can release the project as a "drop in" to an existing Sakai release. The team can produce install documentation for their product and work with community members who choose to use the new product.

Community adoption is an important phase for any new project to go through. Generally no new project will be accepted into the Sakai release until it has demonstrated that it works effectively and meets user needs at a number of community sites. It is also important that when members of the community other than the initial developers can install and run the product reliably in production, it gives some assurance that the software will not be a problem when it is made part of the Sakai release.

Provisional Status
Once a project has been successfully adopted and used by some subset of the community, it can be considered for provisional inclusion in a Sakai release. A provisional capability in Sakai is one that is included in the release but "turned off" by default.

During the release and QA process, it is up to the respective Project team to do QA on the provisional project. The Sakai Foundation QA effort focuses their effort on the

components that are fully supported in the release. The Foundation QA effort, however, does test to see if the introduction of the provisional project destabilizes the other elements of the release in any way.

There is a series of technical criteria a project meets to be considered as provisional. These are described in detail in a separate document. The high level overview is that a provisional tool is technically and legally part of the Sakai release - this means that things like licensing and contributor agreements must be 100% completed and 100% acceptable to the Sakai Foundation even before a technical analysis of the project is undertaken. This suggests that projects intending to be part of the release should start dealing with contribution agreements and licensing issues very early in the project.

The project team for a provisional tool must be willing to participate in and support the Sakai release process, QA effort, bugs tracking, etc.

Sakai's Provisional Status is *somewhat* similar to Apache's Incubator status in that they both describe "projects in waiting" or "on probation". But again, the detail here is left to a separate document.

Apache Reference: http://www.apache.org/foundation/how-it-works.html#incubator

Full Component of the Sakai Release

Once a tool has been a provisional tool and appears to work well within the Sakai release, it can be promoted to being part of the official release.

There is a set of technical and other requirements that a tool must meet to become part of the Sakai release. The provisional release requirements "bar" is set somewhat low to encourage new tools to make it in as provisional tools. However, the "bar" is higher for full components.

Additional elements required for promotion to full release status include: test plans, accessibility audit, internationalization support, import/export support, and others. The high level summary is that to be a Full Component in a Sakai release the new element must function like the rest of the Sakai tools and be fully and seamlessly integrated into Sakai.

Guiding Principles

There are a number of guiding principles that underlie this document. This section attempts to capture the nature of these concepts.

Constant Communication

This is the foundation of informed collaboration between intelligent people, and in a community of cooperating projects, is the key to understanding and effectively pursuing any and all activities within that community. It is a requirement for an organization that

must bring together a variety of complex components into a working whole, and reflects our experience that successful volunteer collaborations are built on the contributions of informed individuals who have common goals but often wildly different styles of working and problem solving. Constant communication keeps the necessary descriptions and explanations flowing and available to everyone.

We consider this one of the principles that should be followed in all the various venues and efforts that make up the Sakai Community.

Meritocracy

From Apache: We call this basic principle "meritocracy": literally, governance by merit.

Decisions and directions in Sakai are made by those who have demonstrated "merit" rather than those who are in a "position" or have a "title". The Sakai Foundation staff members described above have responsibility for the overall effectiveness of the project and will likely be very influential in Sakai's directions. But in all cases, they have no granted authority which overrides the authority of the PMC's. The Sakai Foundation staff must earn their "merit" like any other member of the Sakai Community.

From Apache: What is interesting to note is that the process scaled very well without creating friction, because unlike in other situations where power is a scarce and conservative resource, in the apache group newcomers were seen as volunteers that wanted to help, rather than people that wanted to steal a position.

Reference: http://www.apache.org/foundation/how-it-works.html#meritocracy

Openness

While PMCs in Sakai are given broad powers to control their own destiny, it is critical for them to do their work in the open. Without openness, there is no way to harness the talent of the entire community and no good way for new members of the community to come up to speed so they can contribute.

From Apache: We endeavour to conduct as much discussion in public as possible. This encourages openness, provides a public record, and stimulates the broader community. However sometimes internal private mail lists are necessary. You must never divulge such information in public without the express permission of the list. Also never copy an email between private and public lists (no Cc). Such an event would go beyond the normal need for email ettiquette and be a serious breach of confidence. It could have serious ramifications, cause unneccessary confusion and ill-informed discussion.

Reference: http://www.apache.org/foundation/how-it-works.html#management

Understanding the Nature of Volunteer Resources

Other than the Sakai Foundation staff members, everyone in the Sakai Community is a volunteer. While they may be paid to work for some company or university that is involved with Sakai, strictly speaking the Sakai Foundation must treat individuals as volunteers.

Respecting the volunteer nature of the community is why there is very loose coupling between the projects and the Sakai Foundation. The Sakai Foundation can communicate, suggest, guide, produce requirement documents, identify areas that are high priority, do roadmaps, and even drop a software element from a release if it is not technically ready. But through all of this the Foundation must respect that it cannot give "orders" to the volunteers that make up the projects.

If you want to affect a project - Join it and Contribute

Many projects will be very busy and working very hard on a long series of tasks that stretch out for possibly many years. The members of a PMC are the ones who best know the complexities of the work laid out in front of them. It is a mistake to believe that the Sakai Foundation staff or even members of the community will naturally know what is best for any given project.

The only way to truly have impact in a project is to take the time and find a way to contribute to the project. Simply standing back and telling a project what is the best way for it to do its work, or criticizing the project is generally a poor way to have impact.

The Right to Ignore

A natural effect of working in the open is that there will often be far more users/lurkers than PMC members. The PMC uses the mailing list to work through issues and make their decisions in the open.

Simply joining a list or being a user does not give a person the right to "vote" or affect direction. This is done over time by becoming a developer, committer, and eventually PMC member.

While open dialog, criticism, and brainstorming on the project list can be quite useful; the PMC usually has some task and work that needs to be done.

If a user/lurker jumps into a conversation and sends messages that are not on topic or not particularly useful, they should not be surprised if their message is simply ignored by the PMC members. If a user or developer wants to help, the best approach is to listen carefully for a while, understand the issues and then find a way to help.

If a user or developer persists in pursuing a conversation to the point that the discussion is disruptive to the operation of the PMC, that user will be asked to move their discussion somewhere other than the PMC's mailing list. In Sakai we have a list called OpenForum

where any topic can be debated and discussed as long as folks find it interesting. Project lists are there to allow the PMC to do their work in a public way.

Lack of Need to Always Agree

The Sakai Foundation strives to create an environment where decisions are made by those best placed to make them, in the open, by those "on the ground" and contributing to the solution, and with as much relevant consensus as possible. The Sakai Foundation also works to create an environment where disagreement can flourish without being crippling, where alternative solutions are supported, and where, in the most cases possible, we can "agree to disagree" and the work can go forward.

Our community is rooted in intellectually and technically innovating organizations that are very diverse along a very large number of dimensions. The Sakai software, its community and its goals attract individuals that work in rapidly changing environments and are working hard to accelerate that innovation and change. In many cases the creative opportunities that current design methods, software tools and development environments provide allow for a wide range of approaches to solving particular needs. In as many places as possible the Sakai way is to support alternative approaches which can be locally realized to best effect. It tries to find ways where "we don't need to agree" on the detail, and multiple approaches can be taken in the projects doing the work.

Conflict Resolution

If the Apache experience is any indication, these procedures, approaches and values will avoid conflict. Given that projects and the community are charged with working together as mutually respecting peers, hopefully mutually agreeable solutions will be worked out at the lowest possible level and not require any involvement "from above". Hopefully all projects take as a founding principle to work in the best interest of the community and not in the personal interest of the PMC members.

However, the Sakai Foundation Board does retain the ultimate ownership of, and responsibility for, these projects within Sakai. The Foundation chooses to grant this authority to the PMC's for each project with very few "strings attached" - but the Foundation does remain, ultimately, responsible for the successful operation of the projects.

If there is a conflict between two projects, or between a project and the community, the Sakai Foundation Board and Staff will primarily try to mediate the debate and get the parties to communicate better, then perhaps understand the nature of the conflict, and then perhaps help the parties to find a mutually agreeable solution.

If a project and PMC seems to have a pattern of not operating in the community's best interest, or has a history of unreasonable uncooperative behavior, the Foundation may step in and take explicit action. This action will likely take one of the following two forms.

- The removal of the PMC Chair (Lead committer) or the complete removal of the entire PMC with the hope of reforming a PMC that will operate in the best interest of the Sakai community and Foundation.

- "Demoting" the project from release status to provisional, or possibly even independent contribution status, with the hope that a replacement project will grow to fill the space left by the removal of the project from the Sakai release.

These should be understood as drastic actions, only applicable to extreme cases, and taken only after it seems that every avenue to resolving the conflict has been tried. In the Apache experience, with well over 20 active projects, this type of drastic action happens less than once per year.