Introduction

QA process improvement: The underlying changes in supporting technology are less important than general process improvement. The Foundation is already looking at a continuous build infrastructure this is an excellent starting point for improvement. However, the role of QA is not just about removing defects, but also to support decision makers in judging risks and test management for others to reuse tests locally and view historical progress.

Discussion

The sooner you catch a bug in the software cycle the cheaper it is to repair. We should promote continuous builds (not just one build for trunk) environment with the continuous builds having its own branch like manager. Any code that breaks the builds should be considered smelly and be reviewed by a pool of peers.

The QA process should not demotivate or ignore developers or potential testers. A clear process, a structured set of online content and the ability to give feedback and be listened to is important. One could imagine weekly reports being sent by email to interested parties and archived on a static website. The executive director of Sakai should be able to read a short summary and know where the product range is in its lifecycle. The summary should be automatically updated at least once a week. Further, the product council need to have access to reports that detail metrics that are helpful for scorecards.

Anything with an official Sakai tag on it needs to have a neutral process measuring its quality and defining the entry level for acceptance. *I expect scope to be a matter of negotiation.* The scope of the continuous build environment should include the active branches, 3akai and any contributed project that are trying to pass through the product council's definition of quality

We should act small and think big. Starting with the deploying of one automatically tested QA server, we should improve and then deploy on all QA instances under a wide range of infrastructural conditions and code bases. The code bases can be systematically swapped between different server OS/database types.

Due to the size of the code base, we should define quality zones where code such as the Kernel is most thoroughly tested and built more often than a particular contrib project.

The QA should be most focussed on the code that has changed since the last cycle than on a full regression set.

To capture design and performance issues earlier in the cycle periodic human tests on a QA server under load by a Functional administrator (someone who understands the requirements such as a product council member) should take place. A tester should never test their own project. This implies that personal mostly involved in project x can test project Y and project Y personal project X.

We should look at automatic stress and functional testing as part of the continuous build process, only if the cost of implementation is paid back by active use. There is no point in building

infrastructure that does not support decision makers or generate a clear set of to-do's for developers that they then enact.

There needs to be a clear process resulting in the removal of potential bugs early. We should reconsider the role of static code review and code review and look at performing both just before a code freeze, a developer pool should then clean up the list.

There are islands of excellence within the community for various parts of the QA process. Examples include performance testing, automatic functional tests, writing well coded unit tests, best practices etc. The Foundation should facilitate partnerships with those islands and specific projects.

We should seek ways to gather and understand real world usage of Sakai and playing realistic usage patterns within the test environment.

We should be critically honest, specific projects are weaker than others and may need a small taskforce to bring them up to the Sakai norms, not necessarily improving the code base, but rather improving the local QA process.

Are we inclusive enough in the QA process are there potential QA testers either put off by the structure of the confluence site or a feeling of complexity? Should we reach out more to less traditional help?

We should continue to reuse test scripts and build a simple to understand test management structure so that anyone in the community has clear access. One could imagine a static website pointing to a wide range of test scripts and their historic results in subversion. The Foundation should seek to harvest from local QA and expose best practices to a wider audience.

The community should look at a QA profile in the main pom.xml of Sakai where deployment of a localhost test server occurs with specific multithreaded tests run and new random tests pulled out of subversion.

As the Sakai experience moves from Sakai as a set of tools to Sakai as a set of services rendered by widgets the quality of the client side code will become crucial to Sakai's adoption. For example, we should start to enforce minimum coding rules in JavaScript and look more critically at automatic security penetration tests.

There are significant Quality Assurance issues as the community tries to maintain 2.x and migrate to 3.x and onwards. Migration routes need to be clearly defined and thoroughly tested against a wealth of real world data. This requires the development of new methodologies and thus the functionality should be developed early in the lifecycle, for example Sakai migration needs a clear agreement on import and export formats.

PLAN

Phase 1

- Define scope of QA and discuss with the audience within that scope. For example, is Cambridge going to share QA infrastructure? Draw diagram of scope, infrastructure and responsible parties to make it clear to the wider community what are the relationships and responsibilities lie.
- Collect local QA resources from community and develop a simple QA test management website backed onto subversion.
- Improve non technological processes
- Start pushing automated weekly status reports
- Write detailed recommendations and action plan
- **[Review]** Improve based on feedback from Foundation staff

Phase 2

- Setup basic continuous build server
- Modify pom.xml for QA profile for testing and site generation
- **[Review]** Improve based on feedback from Foundation staff
- [Expand] number of QA servers and code bases involved

Phase 3

- Enact small technological projects aimed at generating action lists and/or improving communication