Software Sustainability Services

Some vague initial technical thoughts

Chris Cannam, March 2010

What services do we want to offer?

- Code repository
  - For use by developers of new software and maintainers of old software
- Build service
  - Continuous integration
  - Cross-platform builds, unit tests
- Data service
  - Perhaps integrated with build & test procedures
- What else?

Successful services

- Will be immediately useful
  - Obvious how the service will assist research & development
  - Factors specific to our field to make it more compelling than other existing services
- Will not present cultural or diplomatic obstacles
  - *e.g.* support “private” work protected (from us, and others) by both technical and policy mechanisms
- Will help to build the community by making associations between related pieces of work, and between code and publications
- Will be friendly, reliable, and trustworthy

Some technical goals

- Simple integrated build service: commit code ⟶ trigger cross-platform rebuild and test ⟶ read a report of current status on the website
- Easy authentication, preferably permitting existing credentials such as OpenID
  - At least 15% of our Subversion users have forgotten their passwords
- Support multiple projects with different “owners”, but permit individuals to work on many of them
- Support various “hosting levels” (see next page)

Levels of hosted-ness

1. New code developed entirely within the service
   - Open source / closed source / “decide later”
2. Code published elsewhere then maintained within the service (abandonware, forks)
3. Code published elsewhere but mirrored / tracked within the service (build, test, portability services)
   - Diplomatic considerations – needs to be clear that this is an assistance rather than a rival to existing developers
4. Code hosted entirely elsewhere but which the service assists indexing and discovery of?

Things that would obviously be good

- Providing everything we would need to maintain Sonic Visualiser (current build infrastructure in particular is a bit of a mess)
- Providing build, test, and general help infrastructure for prospective authors of Vamp plugins
- Providing whatever would be useful for e.g. MSc students trying to convert Xue's code to plugins
- What else do we know about already...?

Review of existing “forge” services

- SourceForge → the original, recently overhauled
- Savannah, BerliOS, Alioth, many others → based on old SourceForge code
- Google Code → simple but not always intuitive; popular with the true geek
- GitHub, BitBucket → specific to Git and Mercurial respectively; former is very trendy at the moment
- Lighthouse, Assembla, JIRA → commercial offerings aimed at business use
- Launchpad → Ubuntu/Canonical hosted service

Software for running your “forge”

Name	Comments	For	Against
FusionForge, Savane	Forks of “original” SourceForge code	Widely used	Clumsy compared with newer software
Trac	Minimal, developer-focused code and tracker application	Widely used	Single project per instance; not intuitive to normal people
InDefero	Clone of Google Code		
Origo	Academic project? Published by ETH Zurich		Ugly; apparently not widely used
Redmine	Uses Ruby on Rails	Nice to use, has Mercurial support	
Retrospectiva	Reminiscent of GitHub; uses Ruby on Rails		

Version control systems

- Centralised
  - Subversion the only current practical option
- Distributed
  - Git and Mercurial most popular, Bzr most unpronounceable
  - Git popular with kernel hackers and überleet
  - Mercurial more limited but easier to get into, better cross-platform support, possibly easier to manage server for (authentication etc)
- Centralised or distributed?
  - Distributed may be far more appropriate when hosting or mirroring software that was originated elsewhere
- Single system, or a choice?

Continuous integration systems

- BuildBot (Python)
- Hudson (Java)
- TeamCity (primarily commercial)
- CruiseControl
  - Originally Java, Ruby implementation (CruiseControl.rb) looks promising if using one of the Ruby forge systems