Distributed revision control

Latest update: At TikiFest CEST 2015, the topic of switching to GIT was again discussed. In general, since the discussions a couple of years ago there is increased interest in doing so, especially since most modern projects have switched, and also there is more knowledge of how GIT works and its benefits. See What ToDo When Migrating To Git for more information on the preparatory work to plan for such a transition. In the meantime, if you want to simply use GIT with Tiki, there is a GIT mirror of Tiki's SVN repository which is updated hourly. See Using Git with Tiki for more information about that mirror.

This page is to discuss and if decided, coordinate the eventual transition to a Distributed revision control system (DRCS). Tiki started in 2002 with CVS and switched to SVN between 1.9 and 2.0 The current situation with SVN is good and there is no emergency to change. This is just to explore any new opportunities.

We have **not decided to switch** to DRCS and we have not decided **not to switch**. If ever we do switch, the software is not picked. So for the forseeable future, **all developers interested in using Git should go right ahead and use it and commit to the main repository using git-svn**. The best of both worlds!

We should take the time to pick the best system for **our** model. Sometimes, a DRCS is introduced to provide more flexibility to "non-core developers". In Tiki, since all devs have commit access to the whole code base, this doesn't provide us with the same "advantage" as it would another project.

Rodrigo Primo wrote:

> *I have been using a local copy of Tiki repository created with git-svn for almost two years now. It is not the same as having the whole project using Git, but in my opinion, for development purposes, is much better than SVN. You can use most of the nice Git features like git blame, git stash, git bisect just to mention a few.*

Why

- Could ease the management of Experimental Branches and Quality Team
- Easier merges (even if upward merges are less and less used)
- Perhaps would help with Translation branching strategy
- Would help with Configuration Management and Systems Orchestration

Why not

- Developers (such as Rodrigo) can use and benefit from Git and commit to SVN, without affecting the workflow of everyone else (we have 500 accounts on SourceForge.net)
- Is the effort worth it?
    - It's a lot more work than many people may think. Tiki is a large project, and there are many scripts and setups. The migration to Allura should have been very simple because it's' mostly just about changing the repo location, but it caused extra work for developers.
- Is it harder to use than SVN?
    - With SVN, you need to keep in mind 3 versions of a page:
        - BASE version (i.e. the version that was in the repository last time you updated it)

        - LOCAL version (i.e. the version you have on your machine)
        - HEAD version (i.e. the version that is at the HEAD of the SVN repository)
    - ~~With a DVCS, you have to **keep in mind 5 versions**:~~
        - ~~LOCAL BASE version (i.e. the version that was in the LOCAL repository last time you updated it)~~
        - ~~LOCAL LOCAL version (i.e. the version you have on your machine)~~
        - ~~LOCAL HEAD version (i.e. the version that is at the HEAD of the LOCAL repository)~~
        - ~~REMOTE BASE (i.e. the version that was on the remote repository the last time you imported it to your local repository)~~
        - ~~REMOTE HEAD (i.e. the version that is at the HEAD of the LOCAL repository)~~
    - this is simply not true...there are plenty of information showing that branching and merging are easier in DVCS than with SVN, e.g. this post. Moreover there are DVCS like Fossil which have so called *autosync* mode which brings the best of both worlds. For a proof you can just inspect how does Fossil's timeline looks like - almost linear with easy branching and merging and **without** rebase providing safety not to lose anything 😉

Minimum criteria

- Needs to be offered by SourceForge.net
    - We have 500 accounts there
    - ~~The features implemented/offered by SourceForge.net should fit our needs~~
    - Moving to GitHub or a self-hostes GitLab isn't out of the question any more, especially since sf.net has been perceived as less stable and trustworthy in recent years
- Need a read-only from SVN so shared hosting environments can easily update
    - Even better if commits from SVN are possible
- Need a multi-platform client because we have developers on each.
- Moving away from SVN means updating the release scripts, TRIM and our Continuous integration infrastructure, which is significant work.
    - Thus, the person that leads the migration efforts would also take the lead to update all the release scripts.
        - A beta of the new release scripts should be available long before the official migration to the new system, because we need to make sure not to delay the release.
            - This person will be the package manager and be responsible for the merging process until it's stable.

Nice

- Help address other needs such as
    - Web commits
        - This would increase collaboration from translators and theme designers
    - Improving Code Review workflow to reduce time investment of the Quality Team
        - So tool should have code review or let us integrate with code.tiki.org
    - Continuous Integration
    - Ease Version lifecycle (maintenance of 4 active branches)
        - Better merging would be nice, especially translations
- As much support as possible from integrated development environments (such as XAMPP-Aptana) (Eclipse

has a Subversion plugin)
- As much support as possible from operating systems (Windows has TortoiseSVN, GNU/Linux has kdesvn for KDE)

If so, when

If so, which one?

Git

GIT seems to be a powerful alternative to SVN and there are some plans to integrate it into the current development system. We don't want to migrate to git, but rather mirror svn in git. See - https://github.com/alloy/git-svn-mirror

Linus on git

http://sourceforge.net/apps/trac/sourceforge/wiki/Git
http://gitlabhq.com/

A plugin exists for the Aptana Studio IDE

Bazaar

- http://sourceforge.net/apps/trac/sourceforge/wiki/Bazaar
- A plugin exists for Eclipse, but not for NetBeans IDE.
- https://launchpad.net/bzr-svn
- http://wiki.bazaar.canonical.com/BzrForSVNUsers
- it's almost dead, iow. no more development - see this post

Mercurial

- http://sourceforge.net/apps/trac/sourceforge/wiki/Mercurial
- A plugin exists for Eclipse, and for NetBeans IDE.

Fossil

- http://www.fossil-scm.org/
- http://irc.tiki.org/irclogger_log/tikiwiki?date=2012-11-24,Sat&sel=48#l44
- http://fossil-scm.org/index.html/doc/trunk/www/fossil-v-git.wiki
- http://mail-index.netbsd.org/tech-repository/2010/01/18/msg000350.html
- http://bsdtalk.blogspot.ca/2010/07/bsdtalk194-fossil-scm-with-d-richard.html

Android ports

    - https://play.google.com/store/apps/details?id=es.dadbiz.fossil
    - http://android.wanderinghorse.net/fossilite/
        - http://android.wanderinghorse.net/repos/fossilite/index.cgi/index

Guis

- http://www.tortoisefossil.org/ (Will be like TortoiseSVN, TortoiseGit, etc.)
- https://code.google.com/p/fuel-scm/ (cross platform)
- http://repository.mobile-developers.de/cgi-bin/ikoch/sharpfossil/wiki?name=WinFossil
- https://code.google.com/p/jurassic-fossil/

Questions

- How to deal with externals from Git or SVN?

Pros

- Simple to deploy (one file for the app, one file per repository)
- Works on shared hosting
- Fossil Bisect

Cons

- Not well-known
- Web-based file editing aka Web commits is on the todo list
- Not yet supported by Ohloh.net
    - http://www.fossil-scm.org/index.html/tktview?name=0f36ec7790
- Not yet supported by Allura (All SourceForge.net hosted projects will eventually be moved to Allura)
    - http://www.fossil-scm.org/index.html/tktview?name=311671db59

Monotone

- http://www.monotone.ca/
- https://en.wikipedia.org/wiki/Monotone_(software)
- https://www.ohloh.net/forums/8/topics/504

Related

- https://blogs.atlassian.com/2013/01/svn-to-git-how-atlassian-made-the-switch-without-sacrificing-active-development/
- http://redmonk.com/sograd y/2012/11/05/dvcs-2012/
- http://www.flourish.org/blog/?p=397
- The Risks of Distributed Version Control
- Choosing a distributed VCS for the Python project
- Distributed Version Control Systems - Why and How
- Mercurial vs Bazaar vs Git
- https://www.ohloh.net/tags/dvcs
- Git or SVN? (for Zend Framework 2.0)
- http://better-scm.berlios.de/comparison/comparison.html
- Lessons from PostgreSQL's Git transition

- MediaWiki & Git
    - http://blog.wikimedia.org/2012/02/15/wikimedia-engineering-moving-from-subversion-to-git/
- OGRE DVCS decision
- http://googlecode.blogspot.com/2009/04/mercurial-support-for-project-hosting.html
- http://google-opensource.blogspot.com/2011/07/announcing-git-support-for-google-code.html
- http://justaddwater.dk/2009/03/09/using-git-for-svn-repositories-workflow/
- http://google-opensource.blogspot.ca/2008/05/develop-with-git-on-google-code-project.html
- https://github.com/changi67/tiki/
- http://scottchacon.com/2011/08/31/github-flow.html

alias

- Distributed Version Control Systems
- DVCS
- Mercurial
- Bazaar
- Fossil
- Monotone