Subversive Cross-Team Componentization

Andrew Ferrier

MQ & ESB

May 2004

Abstract

Use of Subversion, an open-source source code library tool designed to supersede CVS, can provide a stop-gap for when time constraints don't allow full componentization. This could be of use to the IBM development community by maintaining dependencies between different code bases more easily. Subversion enables code changes from one code base to be semi-automatically merged into another code base, whilst the second code base is undergoing change.

Subversion is a stable library system, which also has many other compelling features. There are few reasons not to use it in preference to CVS.

Subversion's Hidden Talent: 'Vendor' Merging

Scenario: There are two products, **Flangifier** and **Grommilator**, which want to share a component **Wonkify**. Wonkify has already almost been finished by Flangifier's development team, although they have some bugs to fix. But there isn't time to make it more abstract so it can be directly used by Gromillator before Gromillator needs to be ready.

Solution: Flangifier's Wonkify has a variety of files. Gromillator's development team check these into their Subversion repository:

```
svn add flangifier/wonkify/*
```

```
flangifier/ [1]
`-- wonkify/ [1]
    |-- java/ [1]
    |   `-- wonkify_api.java [1]
    |-- make/ [1]
    |   `-- make.xml [1]
    |-- wonkify.cpp [1]
    `-- wonkify.h [1]
```

The [1] indicates the last revision during which the file changed.

They then make a copy of it into another directory within the repository via Subversion:

```
svn copy flangifier/wonkify/ gromillator/wonkify/
```

Gromillator's team work on the copy, removing Flangifier-specific features, fixing a few bugs, and changing the component to work with Gromillator.

```
flangifier/ [1]
`-- wonkify/ [1]
    |-- java/ [1]
    |   `-- wonkify_api.java [1]
    |-- make/ [1]
    |   `-- make.xml [1]
    |-- wonkify.cpp [1]
    `-- wonkify.h [1]
gromillator/ [2]
`-- wonkify/ [2]
    |-- java/ [2]
    |   `-- wonkify_api.java [2]
    |-- make/ [2]
    |   `-- make.xml [2]
    |-- wonkify.cpp [2]
    `-- wonkify.h [2]
```

The files in the *flangifier* and *gromillator* directories, once the copy has taken place, are treated totally separately, although internally Subversion stores them as pointers to the same data until one is modified – this is usually described as 'copies are cheap'.

Easy Merging (contd.)

But Flangifier's development team have been working on Wonkify in the meantime – and not only have they fixed a few bugs, they've added some features also. Files and directories have been deleted, added, changed – how can Gromillator's team get up-to-date with these latest changes easily?

Gromillator's team check the latest version of the Flangifier code into the same location as before and then perform a merge:

```
svn_load_dirs $REPOS flangifier/ /tmp/latest_flangifier

svn merge -r1:98 $REPOS/flangifier $REPOS/gromillator
```

```
flangifier/ [98]
`-- wonkify/ [98]
    |-- make/ [98]
    |   `-- make.xml [98]
    |-- wonkify.cpp [98]
    |-- wonkify.h [1]
    `-- wonkify_api.java [98]
gromillator/ [99]
`-- wonkify/ [99]
    |-- make/ [99]
    |   `-- make.xml [99, C]
    |-- wonkify.cpp [99]
    |-- wonkify.h [2]
    `-- wonkify_api.java [99]
```

(note that *wonkify_api.java* has moved, and *wonkify.h* has not changed)

```
flangifier/ [98]
`-- wonkify/ [98]
    |-- make/ [98]
    |   `-- make.xml [98]
    |-- wonkify.cpp [98]
    |-- wonkify.h [1]
    `-- wonkify_api.java [98]
gromillator/ [2]
`-- wonkify/ [2]
    |-- java/ [2]
    |   `-- wonkify_api.java [2]
    |-- make/ [2]
    |   `-- make.xml [2]
    |-- wonkify.cpp [2]
    `-- wonkify.h [2]
```

Subversion automatically merges in all the non-conflicting file and directory changes from Flangifier (with a small amount of help during the *svn_load_dirs* stage). The only manual work they need to do is to identify any files which changed *irreconcilably* in both Flangifier and Gromillator (such as *make.xml* here) and resolve those points using a standard merge tool

Based on an original idea from the Subversion book:
http://svnbook.red-bean.com/en/1.1/svn-book.html#svn-ch-7-sect-4

Uses and Other Features

The above process is slightly simplified, but this basic technique, using Subversion, is being used successfully to keep code bases between two teams in sync. as part of a forthcoming ESB product. Generalizing the codebase would not be feasible in the available timeframe. Nevertheless, this technique does require discipline and is probably be best suited to small development teams – where a particular component needs special handling or as a replacement for CVS.

The process is not new, but it is rarely used, yet straightforward to do, and can be a helpful stop-gap for schedule difficulties. More complex schemes are also possible.

Subversion also has other compelling features as a library system:

- It has a similar interface to CVS – this is familiar to many developers. CVS does some of the above, but doesn't (for example) version directories – vital for a complex code base.
- Compared to CMVC, it is considerably simpler because version numbers are per-repository, not per-file, so a track / commit / version number are all the same concept for Subversion. Also, releases / components / tags / branches are all implemented simply by copying a directory tree.
- It handles binary files efficiently using a binary differencing algorithm.
- The server runs on Linux, Windows, etc. Several graphical clients are also available for various platforms in addition to the command line client.

For more information, visit: http://subversion.tigris.org/