RFC: Using SVN branching to improve software development process at THG

Elena Pourmal, Quincey Koziol

February 9, 2008

Background information

For many years The HDF Group has been developing software using CVS or SVN trunk for the development of the new features, and branches to maintain the released versions of the HDF5 library. In the past few years it became clear that this approach doesn't work well for both new features development and the HDF5 library maintenance.

Work on the 1.8.0 release showed that this approach is OK for the development process when we add new features that do not require much prototyping work (for example, adding new compression like scale-offset or n-bit, or new HL APIs). But for features that require file format changes, or a lot of prototyping to get the best performance (for example, metadata caching work, and work on the compact storage and access by objects' creation order), adding these kinds of features to the trunk causes source code instability and delays the delivery of other stable features to our users.

Additionally, our current practices with the release branches also showed deficiency in addressing HDF5 users' needs. There is no mechanism to provide official patches and keep a record of them. For example, in 1.6 branch documentation files reside inside the source distribution. If we find a bug (say broken link), we can updated our Web pages with the fixed documentation, but we cannot provide official patched release that will have this minor change. Another example will be a patch that addresses some specific platform needs and bugs (VMS or Windows), or specific request from a customer such as a very minor bug fix in the officially released code, or specific compiler problems, or say fix for a data corruption bug that doesn't require a lot of changes to the source. In all cases mentioned above (except probably the last one) no extensive testing is needed.

To address this problem we usually do one of three things:

- Distribute a patch from the FTP site with the instructions how to apply it
- Provide a snapshot directly to a user that may contain more code that was in the official release
- Come up with the new unplanned release

Clearly there are a few problems with this approach:

- There is no "official" record for the patch in SVN
- Many users cannot accept snapshots
- It is hard to stick to the release schedule

We propose to review our SVN practices to address the problems with the current software development and maintenance process.

Proposal

1. *We propose to use "feature" branches for new code development.*
2. *We propose to use "errata" branches to address problems in the release branches in between maintenance releases.*

How does it work?

1. **"Feature" branch**

   Every time we start working on the new feature, a new branch coming out of the trunk is created. It is developer's responsibility to test the code in that branch, to bring back changes from the trunk, and to keep code in sync with other developers. When the feature is completed, tested and *documented,* it can be brought back to the trunk. This approach will give us more stable code in the SVN trunk, allow us to do a better planning for the major releases and provide stable snapshots for customers that work in an agile development environment. This approach will also give us opportunity to share and to review the code under development from the first stages, and have more than one person working on a feature. Developers will need to learn how to work with SVN branching, code syncing, etc. Quincey's "Feature Branching in Subversion" (http://www.hdfgroup.uiuc.edu/internal/QA/software-engineering-at-THG/Feature-Branching-in-Subversion.txt) document addresses those issues. Using feature branches is already underway, with the work on journaling changes to HDF5 metadata proceeding in a feature branch, as is work to update the FORTRAN API wrappers for the new routines in the 1.8.0 release, and others as well.

2. **"Errata" branch**

   Every time we do a release, "maintenance team" creates an "errata" branch off of the release branch, in order to track minor updates to the released files. Only a small subset of the changes that go into the release branch in between maintenance releases will end up in the "errata" branch. The "errata" branch is terminated at the time of the next maintenance release. Testing of the "errata" branch should be done at the time of the next "errata" release and in most cases is limited to one-two platforms.

   Here are some use cases for the "errata" release:

i. TX3 (Red storm) was unavailable for testing at the time of the HDF5 1.8.0 release. “Platforms tested” section indicates that testing was not done on that machine. A few weeks after the release, Red Storm comes on-line, we test HDF5 1.8.0 release code and discover that a compiler flag is needed in order to build and test the library on that machine. Configuration file for Red Storm is modified in both HDF5 1.8 release and errata branch. After testing “errata” distribution on the machine, we release HDF5 1.8.0.1 that has support for XT3.

ii. We have a paying Windows customer who would like us to support multithreaded static builds of the HDF5 library. Our current projects in the source code support only single-threaded or multi-threaded DLLs. We can safely add new projects to “errata” branch, and release HDF5 1.8.0.2

iii. Missing parameter in a Fortran wrapper was reported in a function that is critical to ESDIS application. Bug can be quickly fixed for HDF5 1.8.0.2 and given to the customer.

iv. “=” was used in the IF condition statement instead of “==” causing data corruption in some rare cases. The change is trivial. We add a new test to exercise the bug, and run a daily test script for the branch that tests and builds new binaries. If tests pass, we create HDF5 1.8.0.3 source tar ball and make it available to the users. Newsletter will be send informing users about the changes i) –iv) in the HDF5 1.8.0.3. We also keep binaries and if time permits, we replace binaries on our FTP server or we inform users that updated binaries are available on a request.