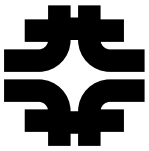

Fermi National Accelerator Laboratory

FERMILAB-Conf-98/301

Software Management at Fermilab

Robert M. Harris

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

October 1998

Published Proceedings of *CHEP '98: Computing in High Energy and Nuclear Physics*, Chicago, Illinois, August 31-September 4

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CH03000 with the United States Department of Energy

Fermilab-Conf-98-301
September 29, 1998

Software Management at Fermilab

Robert M. Harris
Fermilab Computing Division

Presenting for:
Dave Adams, Michael Aivazis, Jim Bellinger, Walter Brown, Glenn Cooper, Flavia Donno-Raffaelli, Lynn Garren, Herb Greenlee, Robert Harris, Alan Jonckheere, Robert Kennedy, Arthur Kreymer, Qizhong Li, Don Petravick, Ruth Pordes, Lars Rasmussen, Elizabeth Sexton-Kennedy, Scott Snyder and Gordon Watts.

Abstract

We describe the structure and performance of a software management system in wide use at Fermilab. The system provides software version control with Concurrent Versions System (CVS) configured in a client-server mode. Management and building of software is provided by Software Release Tools (SoftRelTools) originally developed by the BaBar collaboration. Support for SoftRelTools, the heart of the system, is organized by the Fermilab computing division in close communication with the end users: CDF, D0, BTeV and CMS. Unix Product Support (UPS) is used to initialize environmental variables for multiple versions of software on multiple platforms. Distribution of frozen releases is currently handled by internally developed scripts, but will soon be performed by Unix Product Distribution (UPD). At CDF the development version of the software is also distributed daily and built in place on 18 different machines, with new machines added weekly. Although primarily intended for UNIX platforms, including Linux, the system is also supported for Windows NT by D0.

This system handles the version control, management, building, and distribution of code written in Fortran, C, and C++. A single executable can call routines written in all three languages. A distinguishing feature of the system is its ability to allow rapid asynchronous development of package versions, which can be easily integrated into complete consistent releases of the entire offline software. Daily rebuilds of all the software, along with automatic mailings of build errors to developers, test robustness and allow speedy integration.

This system has been used since January 1997 by CDF, D0 and BTeV for the development and release of software for the next run of the Tevatron Collider. At CDF it has been used by roughly 30 developers to make over a dozen frozen releases of a million lines of software. D0's use is similar to CDF, and the system is just beginning to be used by CMS. The cooperative maintenance and management of SoftRelTools, led by the Fermilab computing division with the active participation of CDF, D0, BTeV and CMS, is discussed as a model for the sharing of common tools in High Energy Physics.

1 Configuration Management Working Group

In early 1996 the Fermilab computing division (CD) formed a group to plan for configuration management in Collider Run 2, currently scheduled for Spring of 2000. The group consisted of members of CD, and the CDF and D0 experiments. Our charge was to find and implement a common CDF/D0 solution for software management including the following: version control (code repository and history tracking), organization (packages and releases), building (tools to compile and link), distribution (sending code to remote sites) and the flexibility to automate testing as necessary. We believe this project was successful for three reasons. First, rather than reinventing the wheel, we adopted standard technology currently in use by HEP: CVS with enhancements from SDSS has been used for four years, SoftRelTools from the BaBar experiment [1] has been used for three years, and UPS/UPD from CD has been used for seven years. Secondly, we work continually on supporting a common system for all experiments. Finally, we listen to our users and tailor the system to their needs.

2 Archival and Version Control

2.1 SDSS-CVS

Concurrent Versions System (CVS) is widely used in HEP. Our configuration for the use of CVS was first implemented by the Sloan Digital Sky Survey (SDSS). We run CVS in client-server mode, with the repository on the server machine, and the client setting the environmental variable CVSROOT to point at a special pseudo-account on the server machine. To access the repository where SoftRelTools is kept at Fermilab, the CVSROOT is set to cvsuser@cdfsga.fnal.gov:/usr/people/cvsuser/repository, where cvsuser is the pseudo-account. The pseudo-account runs a restricted login shell, CVSH, which only allows cvs commands so the user cannot inadvertently modify the CVS repository directly through UNIX commands. The .rhosts file controls read access, and a special script called check_access, invoked during cvs commit, controls who has write access for a given package. Small modifications to CVS at the server end trap the user's host name and user name and insure proper history information. Some of the advantages of this mode of using CVS are that local and remote access are identical, a user does not need to have an account on the server machine to access the CVS repository, and a user cannot accidentally delete or change the protection of files in the CVS repository. Further information and support is available on request from cvs-support@fnal.

2.2 Fermilab Experience with CVS

Our experience with CVS is that it is fast and reliable. Concurrent development is quick and efficient, collisions among developers are rare. We use the loginfo script to send e-mail to package developers whenever their package (cvs module) is modified. This keeps developers informed of changes and enhances the cooperative spirit of code development. The same list of developers can receive build errors from the

Figure 1: Left: Production release consisting of packages pkgA and pkgB. Right: A user's test release and how it relates to the official release. Here pkgB is being developed locally and other packages are taken from the official release.

3.2 Frozen and Development Releases

In frozen releases the software doesn't change. They contain CVS tagged and exported packages (e.g. V01-00-00 of pkgA) and each frozen release is numbered (e.g. 1.0.0), as shown in Figure 1. In development releases the software changes daily. The packages are checked out of CVS and updated daily, the packages are not tagged and the release can be just called development without a number. Experiments at Fermilab decide whether they want daily development releases or not. CDF has a daily development release and frozen release roughly every two months. D0 has weekly frozen releases. BTeV has a daily development release and frozen releases roughly every three momths. Development has the advantage of daily integration. All software is built together everyday, and problems are spotted and fixed quickly. Frozen releases have the advantage of stability. Many frozen releases allow developers to go back to working, but relatively recent software. This temporarily shields developers from problems at some cost to the release manager.

3.3 Tools for Releases and Development

SoftRelTools comes with many tools for release managers and developers. For a release manger, *newver* will export a tagged version of a package from the CVS repository to the packages area and declare it to UPS. The command *newrel -p* will create a production release from a list of package version numbers input by the release manager. For developers, *newrel -t* will create a test release in the user's area for developing, compiling and linking code against a production release (a test release is shown in Figure 1). The command *addpkg* will checkout a package from CVS and add it to the test release for developing code, and *depend* will find other packages that include files in your package and which may also need to be recompiled. Finally, SoftRelTools provides the GNUmakefiles that build the software with GNU Make. All rules are defined, so the developer only needs to specify which executables to link.

3.4 Test Releases

The user environment for developing code is called a test release. In Figure 1, a test release allows the user to develop a package (e.g. pkgB) in the context of a complete production release. The user creates the test release using *newrel -t*, gets pkgB using *addpkg*, and is able to change and rebuild pkgB locally. Any needed header files and libraries of other packages that are needed to rebuild pkgB are then taken from the production release, but all code for pkgB comes from the user's test release. Figure 2 shows the actual commands a CDF user would type to develop the tracking package in a test release. At CDF, the users changes would then appear in development releases on 17 different machines the next morning, which brings us to error reporting.

```
% source ~cdfsoft/cdf2.cshrc        (global initialization)
% setup cdfsoft2 development        (setup development)
% newrel -t development testrel     (create test release)
% cd testrel                        (top level)
% addpkg -h Tracking                (checkout Tracking)
% cd Tracking/Tracking              (Tracking headers)
% emacs Tracking.h                  (edit header file)
% cd ../..                          (top level)
% depend -f Tracking.h              (dependent packages?)
% addpkg -h TrackingMods            (yes, this one uses it!)
% gmake                             (compile and link all)
% TrackingExample.exe               (run to test changes)
% cvs update -A Tracking            (concurrent changes?)
% cvs commit Tracking               (put back in cvs)
```

Platform	Lib Errors	Bin Errors
irix6 - kai	None	Examples
Linux2 - kai	None	TrackingBenchmarks
OSF1 - kai	TrackingMods	Examples, TrackingBenchmarks

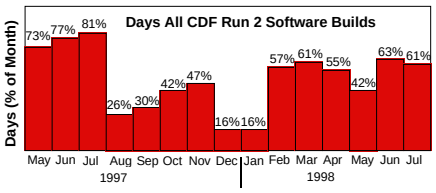

Figure 2: Left: A session where a user develops the CDF tracking package in a test release. Right: Above is a daily table constructed by the error reporter summarizing the status of building CDF software, and below is the fraction of days in each month in which all the CDF Run 2 software compiled and linked using SoftRelTools.

3.5 Error Reporting for Daily Builds

When a compilation or linking error occurs in development it appears in the daily log file. At CDF these log files are collected for each platform (Linux, IRIX and OSF) and scanned for errors by the SoftRelTools error reporter. Errors found within the compilation (lib stage) or linking (bin stage) of a package are mailed to all developers responsible for that package. As shown in Figure 2, a daily table summarizing which packages had lib or bin errors on which platform is constructed by the error reporter and posted on the WWW. The daily building and reporting of errors allows the developers to realistically test their code in the actual full offline environment and bugs are spotted and fixed fast, usually the same day they are caught and reported. As shown in Figure 2, all CDF code builds without any error about 50% of the time: the success rate for any single package is of course much higher. CDF also runs the code and checks for changes in output daily, but this validation is not yet a part of SoftRelTools. CDF finds that daily rebuilding forces them to do integration continually and makes it much easier to produce a frozen release.

3.6 Additions to SoftRelTools by FNAL

SoftRelTools is not a static product. Over the last 21 months the following features were added to SoftRelTools by Fermilab (abridged list):

- Support for Linux, OSF, Solaris and NT operating systems.
- Support for the KAI compiler and its updating of archives all at once.
- Reporting build errors and constructing summary tables.
- New commands *depend*, *lnkpkg*, *newpkg*, and *setcompiler*.
- Refresh mechanism to trigger full rebuilds of libraries.
- Selective builds of sub-packages and single executables.
- Target for test executables, which is not built by default.
- Experiment flag for experiment-dependent parts of SoftRelTools.

Over the next year the following requested changes may be added to SoftRelTools (abridged list):

- Allow *gmake* from subdirectories in addition to top level of releases.
- Implement *gmake debug*, and other options recognizable by SoftRelTools.
- Full support for a hierarchy of sub-packages within packages (CMS request).
- Unification of UPS configurations among experiments.
- Add new targets like validation, clean for single packages, etc.
- Support for shared libraries and automatic link-ordering (PackageList from BaBar).
- Improve test release accuracy when removing files.
- Clean up OS flavor and experiment dependence. Make more modular.

3.7 Software Management on NT

Our system, although desinged for UNIX, was modified by D0 to work under Windows NT [2]. The first version is complete and they are working on improvements. For version control D0 has been using CVS on NT for over a year. The CVS repository is on a SGI server running UNIX, and PC's running NT can access the CVS repository using cygwin32. For releases and building they needed to port SoftRelTools to use the Microsoft C++ compiler under NT. They use the UNIX on NT freeware solution, cygwin32, and translate UNIX commands and switches to NT equivalents. They find that simple make files translate successfully, but ones with weird UNIX commands are more problematic. They also convert makefiles to IDE workspaces, allowing NT developers to use the full NT GUI working environment, while release managers construct releases on NT under the bash shell using cygwin. For setup and distribution UPS/UPD works under NT, although there are still a few oddities being investigated. In the future they hope to have a production quality version of the system.

3.8 HEP-Wide Use of SoftRelTools

SoftRelTools, originating at BaBar, is now used and continually modified by CDF, D0, BTeV, CMS, ATLAS and the Computing Division at Fermilab. BaBar and Fermilab have agreed to diverge but we watch each other's changes for ideas. ATLAS has a modified version with significant structural changes [3], and it may be desirable to merge these into a future HEP-wide common version of SoftRelTools. CDF, D0, BTeV and CMS have access to the CVS repository at Fermilab where our SoftRelTools is kept. CDF builds daily using the head of SoftRelTools, while D0 updates it's frozen version of SoftRelTools frequently, BTeV seldom needs to modify its frozen version, and CMS is just getting started with SoftRelTools. Each experiment is responsible for testing changes, and there is an overal SoftRelTools coordinator at Fermilab who tests global changes. CDF and D0 are the most frequent modifiers, and we catch each other's bugs and benefit from each others improvements, additions and ports. In the last 21 months there have been 777 *cvs commits* by roughly 11 developers. We have a mailing list, SoftRelTools@fnal.gov, which is quite lively and responsive.

4 Setup and Distribution

4.1 UPS/UPD

There have been significant efforts by the computing division to support package management for the coming Tevatron run [4]. To setup environmental variables we use UNIX Product Support (UPS) from the CD. This allows easy setup of special environments with a single command (e.g. setup cdfsoft2 development). UPS provides databases of packages, releases, versions and flavors, which is used by both UPS and UPD. UNIX Product Distribution (UPD) installs packages on remote nodes via either pull or push. UPD for run 2 became available recently. CDF and D0 requested it and are working towards using it. D0 currently uses UPD to distribute to two sites and in-house scripts for two other, in addition to four central sites. CDF currently uses in-house scripts to distribute to 31 sites. Both experiments plan to use only UPD soon.

4.2 Distribution at CDF

CDF wrote its own scripts to distribute all code needed for run 2. This includes CDF software, CD software, and external packages like cern, geant, herwig, etc. We now support 31 sites with at least one frozen release (23 Linux, 6 IRIX, and 2 OSF). Development is being updated daily on 18 sites (12 Linux, 4 IRIX, and 2 OSF). The largest demand is from Linux, and there we developed the simplest procedures for distribution. For a frozen release the user sends mail to rharris@fnal.gov to get registered, logs in as root (in the future root access won't be necessary), pulls one script off the WWW, and executes it. This gives the user everything, setting up all necessary accounts, permissions, and software. For the development release the user logs into the cdfsoft account (created by the previous script), fetches another script off the WWW, executes it, and then tells *cron* to build the release daily by submitting a pre-made *cron* job. If there are problems they are sent to our distribution manager, who provides special assistance to distribution sites [5]. We remotely login to fix site specific problems with the software, networking, and security. We automatically add/remove packages from development on all machines. For Linux machines we optionally provide the CDF software on compact disc, and Fermilab provides a specially supported version of the Red Hat 5 release of Linux [6].

5 Lab-Wide Use of System

The combination of CVS and SoftRelTools is being used lab-wide and supported by the computing division. SDSS, CDF, D0 and the ZOOM project for C++ class libraries all use SDSS-CVS, while BTeV uses vanilla CVS. All these projects use SoftRelTools with the exception of SDSS. Table 1 summarizes the lines of code, packages, releases and distribution sites for these Fermilab experiments. Significant amounts of code, in many packages, rebuilt in multiple releases since January 1997, have been distributed to multiple sites. Table 1 indicates that the system is working,

and that we have succeeded in implementing a common tool for the management of software in high energy physics.

Project	Lines	Packages	Releases	Sites
SDSS	2000 k	132	N/A	N/A
CDF	700 k	69	13 + daily	31
D0	1000 k	119	45 weekly	8
BTeV	15 k	3	6 + daily	4
ZOOM	120 k	9	N/A	N/A

Table 1: For each project we list the lines of code, number of packages, number of releases and total number of distribution sites, all serviced by our software management system. N/A indicates not applicable.

6 Conclusions

For run 2 software management, we have adopted tools that have been used succesfully in HEP for years: CVS as used by SDSS, SoftRelTools originally developed by BaBar, and UPS/UPD from the Fermilab computing division. We have implemented, tested, and maintain a complete software management system at Fermilab. CDF, D0 and BTeV use the system, and CMS is beginning to use it. Improvements to the system are ongoing. New flavors of UNIX and some NT support, development tools and error reporting, and improved setup and distribution with UPS/UPD have all been added. HEP-wide efforts on SoftRelTools have been helpful. We welcome other experiments to join the development and maintenance effort as we progress toward the goal of a common software management system for all High Energy Physics. To join, question, or comment, just send mail to SoftRelTools@fnal.gov.

References

[1] B. Jacobsen, "The BaBar Software Release Structure", http://www.slac.stanford.edu/BFROOT/dist/releases/current/SoftRelTools/SoftRelToolIntro.ps

[2] G. Briskin, D. Cutts, G. Watts and R. Zeller, "Software Release Tools on NT", these proceedings.

[3] L. Tuura, "Overview of ATLAS Software Release Tools", these proceedings.

[4] E. Berman, "Software Package Management and Distribution for Run II", these proceedings.

[5] A. Kreymer, "Porting and Supporting the Fermilab CDF Run II Software under Linux", these proceedings.

[6] D. Yocum, "Linux Support at Fermilab", these proceedings.