A Platform Independent Software Management System

Elizabeth Sexton-Kennedy [a],

The FNAL CD/D0/CDF Run II Code Management Working Group [1]

[a] *Fermilab, PO Box 500 MS 318, Batavia Il. 60510*

The CD/D0/CDF Run II Code Management Working Group has proposed and is implementing a platform independent software management system for Run I and Run II code development for the large collider experiments at FNAL. The system provides for code development, building and distribution across most platforms, including all flavors of Unix and NT. It uses CVS configured in a client-server mode, using a shell on the server developed by the SDSS collaboration, for version control. For management and building, it uses the directory structure and(modified) SoftRelTools developed by the BaBar collaboration. SoftRelTools uses CPP for preprocessing and creation of dependency files and GNU Make for building object libraries and executables. Code and binary distribution is handled by the UPS/UPD product developed by the Computer Division at FNAL. The system can handle code written in virtually any language, including Fortran, C and C++. In particular, the system naturally handles include file dependencies introduced by C++. A distinguishing feature of the proposed system is its ability to allow rapid asynchronous development of package versions which can then be easily integrated into complete consistent production releases. This should reduce the time it currently takes for new code, and corrected code to be made available to users. Most functionality of the system is currently present though development continues.

Key words: Configuration management; Code management; Development environments; Free software

[1] Jim Bellinger, Walter Brown, Glenn Cooper, Flavia Donno-Raffaelli Lynn Garren, Herb Greelee, Robert Harris, Alan Joncheere, Qizhong Li-Demarteau, Don Petravick, Ruth Pordes, Elizabeth Sexton-Kennedy

1 Introduction

Over the past decade both CDF and D0 have used proprietary tools for the VMS operating system to manage their software systems. In 1995 the FNAL computing division declared that VMS would not be a supported OS in run 2. This provided us with an excellent reason to re-evaluate our needs and design a new system based on a freely available tool set. The collider experiments and FNAL computing division formed the Run II Code Management Working Group in March of 1996. It was agreed that a more flexible system which supported a more rapid development release cycle was a necessity for both experiments. Also it was discussed that the nature of the physical design layout in C++ made it even more imperative that we manage our dependences better then we had in the past where the majority of the code was written in FORTRAN. A first implementation of this system was available and partially documented by January of 1997.

2 Choice of Tool Set

Software management tools unlike software development tools are used by every member of the collaboration. This means that the tools selected must be freely available and maximally portable. CVS and gmake provided by the Free Software Foundation, SoftRelTools, from BaBar, and UPS/UPD from the FNAL Computing Division, were selected. These tools needed to be integrated. Because all of these tools were developed on UNIX systems, some care had to be taken with regard to NT. The current strategy is to use the bash shell, CVS, and gmake implementation on NT provided by Cygnus.

2.1 CVS in Client Server Mode using the SDSS CVSH Shell

In this configuration of CVS a special account is setup on the machine containing the repository. This account is the only user who can modify repository files; therefore any modifications must be done through remote CVS access, even if you are logged in to the local machine. The login shell of the special account is CVSH. CVSH only understands CVS commands and will refuse to do any others. This feature allows us to give access to the repository to everyone in the collaboration and solves several security problems. Write access is controlled by a check_access script which is run from commit_info when files are commited to CVS. In all transactions CVSH records the actual remote user and remote host names in the history logs. Note that in all of this nothing special needs to be done on the client. Because of this remote institutions can get CVS from the normal network archives and can upgrade when they want as long as the upgrade is backward compatible with the version running on the server.

2.2 SoftRelTools

At the time we were looking for implementations BaBar had been using SoftRelTools for over a year. Some members of the working group had used SoftRelTools to build BaBar's code on our own systems. The atractive features of SoftRelTools for us were:

(i) SoftRelTools understands the distiction between a new version of a package and a new release of an entire project. This allows asyncronous development of packages and releases. A developer can make a new version of a package without affecting a previous release. This is essential for the rapid development phase both collaborations are now in.

(ii) The knowledge of which package versions belong to which release of the project is contained in the file system. This is an advatage over specifing the package versions only in the environment at the time the build is executed.

(iii) In any build of object file libraries it is necessary to know not only which sources of the package were used but what headers files from elsewhere in the system were used. For this reason, the sources for a package are kept in a different subtree then the releases. Release trees are the only place where you will find compiled binaries. This also makes it natural to distribute the binaries in a different way than the source; (see the section on UPD).

(iv) Developers are encouraged to make their own test releases and therefore do some of the integration work themselves. There are tools that allow a developer to find all of the files affected by a change. The devloper then pulls those affected sources into the test release and modifies them where needed. This effort is essential to insure a rappid release schedule. If the release manager is required to do all of the integration work then we know from previous experience that a release will take a minimum of months to put together instead of the desired one or two weeks.

(v) The makefiles themselves are organized in a hierarchical manner. The makefile for the release recursively invokes all of the makefiles in the directory tree under it. The complexities of dependency file generation and pattern rules, for all supported file extensions, are contained in one file, "standard.mk", which is included by all makefiles in the release. For these reasons the makefiles which package librarians must maintain are very simple. This is absolutely required in our environment because we don't expect all developers to become gmake experts. It also means that if we need to tinker with a rule it can be done in one place and everyone will get the change.

2.3 UPS

UPS is primarily a communications interface between a developer and a user of a UPS product. Generally a developer needs to modify the environment of a user in order for the product to function correctly. When a user types "setup <product_name>" the user is gaurenteed by the developer the ability to run any application or link to any library provided by that product. From the developer's point of view, a UPS product requires (i) a root directory for a product and (ii) an optional script that will be sourced when the product is setup. UPS has been used at FNAL for over 5 years, so continuted use provides backward compatibility.

Each project and package within a project release is a "UPS product". In the original system from BaBar each user was asked to source a SRTstartup.csh file. There was a different mechanism for selecting the version of the release and multiple projects on the same system were not supported. Integrating UPS into SoftRelTools has given us more flexability and allowed better access to the relases for those who are not developers. UPS made it easy to write makefiles allowing users to link their applications to a frozen release.

2.4 UPD

Currently FNAL has two mechanisms for distribution. Both are based on tarfiles. One is UPD, the other is anonoumous ftp. The ftp server also has a web server so that tarfiles can be pulled directly from the web. UPD is fine for distributing a single UPS product. Unfortunately it is clumsy at handling groups of UPS products as is needed for our SoftRelTools projects. Our current strategy is to write scripts to work around these deficencies. We are doing this now in an experiment specific way. It is hoped that this can be generalized or that UPD can be changed is such a way as to support this organization.

Another fortunate feature of the SoftRelTools organization is that it makes it natural to place the package source trees in seperate tarfiles. The release tree along with the appropriate binaries are in a different file. A remote developer can then get an initial distribution via tarfiles and from then on choose to get all updates via CVS. Which means the developer doesn't have to wait for a release manager to make a tarfile before the latest version of code can be used in his own remote development.

3 Conclusion

We have developed a software managment system by integrating together a set of previously available tools. This system is extensively documented [2] and is now being tested and used by the CDF ,D0, and BTev collaborations and the FPCLTF group.

[2] http://www-cdf.fnal.gov/offline/code_management/run2_cmgt/run2_cmgt.html