KNOWLEDGE-BASED SOFTWARE MANAGEMENT*

S. Schaffner, M. Bickley, L. Clancy, K. White,
B. Bevins, JLAB Newport News, VA 23606, USA

Abstract

Management of software in a dynamic environment such as is found at Jefferson Lab can be a daunting task. Software development tasks are distributed over a wide range of people with varying skill levels. The machine configuration is constantly changing requiring upgrades to software at both the hardware control level and the operator control level. In order to obtain high quality support from vendor service agreements, which is vital to maintaining 24/7 operations, hardware and software must be kept at industry's current levels. This means that periodic upgrades independent of machine configuration changes must take place. It is often difficult to identify and organize the information needed to guide the process of development, upgrades and enhancements. Dependencies between support software and applications need to be consistently identified to prevent introducing errors during upgrades and to allow adequate testing to be planned and performed. Developers also need access to information regarding compilers, makefiles and organized distribution directories. This paper describes a system under development at Jefferson Lab which will provide software developers and managers this type of information in a timely user-friendly fashion. The current status and future plans for the system will be detailed.

MANAGEMENT ISSUES

In order to ensure reliable operation of the control system, certain pieces of information about the control system software need to be tracked. The obvious information, of course, is the ability to identify critical operational software, know who owns the software, know where the source code for the software resides, and know where the products of the source need to be installed and run.

Perhaps less obvious is the ability to identify and track the software interdependencies. Correct operation of a software application depends on interactions with underlying support software such as the operating system, compiled libraries, scripts, data files, and documentation. Modifications to any of the underlying support software on which a particular software application depends can produce unwanted effects.

Finally, the ability to reproduce exactly the state of a piece of software from some previous point in the past (including the state of the underlying support software at that time) is crucial to reliable operations since any modification to a piece of working software can introduce unwanted effects (i.e., bugs).

* This work was supported by the Department of Energy under contract DE-AC05-84ER-40150.

Users

There are three types of people involved with the control system software. Software managers who are concerned with the tracking issues described above, the software developers who must provide the information needed for tracking, and the software users who are not directly concerned with these issues but whose work can be adversely impacted when the software managers and developers do not maintain adequate control of their software. Our software management system tries to maintain a balance between the tracking needs of management, the creative needs of software developers, and the reliability needs of software users.

Knowledge Base

The knowledge base for our software management system resides in three areas: the configuration database (we are using Oracle), the code repository (we are using CVS), and the file system itself. The configuration database provides much of the information needed by management to track critical operational software, including the software interdependencies. The code repository maintains a copy of the source code stored in such a way that modifications are labeled which provides the ability to recreate a software application from some point in the past. Software development, testing, and execution take place in the file system.

Management tracking information is captured as part of established standards and procedures that the software developers follow as they work in the file system and as they store modifications in the code repository. Much effort has gone into capturing the management tracking information in as unobtrusive and natural a way as possible for the developers. The benefit is that the more mundane and routine aspects of the software life cycle process actually are made easier when well-defined procedures are in place.

USER INTERFACE

The primary focus of our software management system so far has been on the portion used by the software developers. A command line interface has been developed for a tool set that divides the system into six "managers," each one responsible for a different portion of the system. The managers provide tools for software developers to use to create and maintain the file system and code repository parts of their application in a development area and to install and test their software in the production areas of the control system while at the same time capturing and storing crucial tracking information in the Oracle database.

Application Manager

An application is composed of a set of files. The files may be used to compile an executable binary, or to run through an interpreter or Web server. The set of files may or may not also contain data, configuration, log, or documentation files. For the software developer, the Application Manager provides tools to create a directory structure for an application and set up a module for the application in the CVS code repository. In addition, the Application Manager provides tools to create a release of the application for production and to patch a release for bug fixes or upgrades.

The Application Manager also includes tools for developers to use to register applications that they have chosen to develop on their own or for commercially licensed software or shareware. While developers do not have to use the tools provided to create and maintain applications, they are required to use the registration tools to ensure that the configuration database contains all information needed about critical operational software.

Product Manager

A product of an application is simply a file that will be used in some way either by a user, the application itself, or by another application. Some hand-generated files (e.g. source code files typed in with an editor) generally do not fall under the definition of products. The Product Manager provides tools to register these product files. It is not necessary to register all the files that comprise an application, only those that are critical for the execution, installation, or support of an application. Each registered product file is assigned a unique product id that is used in conjunction with a location id as describe below.

Location Manager

Locations are the places (i.e., directories) in the file system where products are created (the source location) or where products will be used (the destination location). As with products, locations are registered and assigned a unique location id. A single product may be identified uniquely in two locations, once in the directory where it is created using the source location id and again in the directory where it will run using the destination location id.

The Location Manager is used to register locations and assign location ids. When an application is created or registered using the Application Manager, a set of standard directories created as part of the process are automatically registered and assigned location ids. It is the responsibility of the developer to register non-standard locations that are created as part of the process of developing an application.

Package Manager

A package defines the actions that need to take place in order to install (or remove) the products of an application. A package consists of an arbitrary number of items. Each item describes what to do with a single product and consists of the location id for the source of the product, the location id for the destination of the product, the product id, an optional product alias in case the product needs to be renamed at it's destination, and the action to be taken with the product.

Actions include copying or linking files and directories, executing scripts or binaries, and displaying documentation files. Package developers are strongly encouraged to provide documentation about the how to use the software as part of the package.

Packages are really very simple state machines that are stored in the database and provide the system with the ability to perform automated software installation procedures. The Package Manager provides tools to allow a software developer to build and test packages.

Support Manager

Sometimes the products of one application are used by a second application either to build the second application or while the second application is running. This type of application is called a support application and its products are generally not installed into a production area but rather installed into the source area of the application using the support.

One of the goals of our system to is to encapsulate the actions needed to install all support software (e.g., compilers, interpreters, licensed software and shareware, and in-house developed support software) through support packages. The Support Manager is used by a developer to install the products of a supporting application in the source area of the application that is using it as well as to capture the association in the database.

The advantages of providing support in this manner are enormous for both developers and managers. For managers, all dependencies for a particular operational critical piece of software can be determined simply by querying the database. For the developer all available support software is easy to find and install and (if the support package developer has followed the rules) documented. In addition, when an application is installed in a production area, its support automatically goes with it including verification that the necessary support applications are available.

Installation Manager

The major functions of the Installation Manager are to create default installation packages, to install and build applications in one or more production areas (using the installation package), to manage the rollback process for newly upgraded software that may have problems, and to install patches and bug fixes for operational software.

Using the tools provided by the Installation Manager, a default installation package can be built. The standard installation package simply performs a build (if needed) and copies or links products to the designated production directory. The developer is free to enhance the standard installation package if needed.

Since there are multiple production areas at Jefferson Lab, the Installation Manager also provides tools for developers to designate default production areas for an application. In addition, the system supports multiple platforms and the Installation Manager provides tools to automatically invoke the correct compiler for an application.

The history of an application's products is maintained in the database. For any span of time it is possible to determine which release of an application is the current production release, which release is the rollback, which release is available for beta testing and which releases are newly installed or obsolete. The entire timeline for an application can also be provided using information in the database.

One of the requirements of our configuration management system is that the production areas are read-only. The only way to install or to patch production software is through the Installation Manager's tools. This requirement is vital to maintain operational reliability and also offers a great incentive for developers to use the system, since they will not be able to install or upgrade operational software unless they do.

One more function of the Installation Manager is to allow a developer to register a particular release as available for beta testing and then to allow a user to register to be a beta tester for that application. The Installation Manager provides tools to automate this process and also to automatically "unregister" beta testers when a release moves from beta testing into production.

CURRENT STATUS AND FUTURE PLANS

The coding for all the managers except for the Installation Manager is complete. Once the Installation Manager is complete the next step is to release the system and spend some time training developers and moving operational software into the system.

Future plans at this point are to provide tools for software managers and to build a graphical user interface to supplement the command line interface.

REFERENCES

[1] M.C. Paulk, "The Capability Maturity Model Guidelines for Improving the Software Process," Carnegie Mellon University, Software Engineering Institute, 1995.

[2] S.K. Schaffner, "A Relational Database Model For Managing Accelerator Control System Software At Jefferson Lab," contributed to 8th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2001), San Jose, California, 27-30 Nov 2001.

[3] R.T. Snodgrass, "Developing Time-Oriented Database Applications in SQL," Morgan Kaufmann Publishers, San Francisco, California, 2000.