Presented at 6th International Conference on Advanced Technology and Particle Physics
Villa Olmo, Italy, 5-9 Oct 1998

SLAC-PUB-9728

BABAR Note # **478**
January 15, 1999

The BABAR Software Architecture and Infrastructure*

G. Cosmo[a]

[a]SLAC, Stanford Linear Accelerator Center,
P.O.Box 4349, 94309 Stanford CA, US

The BABAR experiment has in place since 1995 a software release system (SRT Software Release Tools) based on CVS (Concurrent Version System) which is in common for all the software developed for the experiment, online or offline, simulation or reconstruction. A software release is a snapshot of all BABAR code (online, offline, utilities, scripts, makefiles, etc.). This set of code is tested to work together, and is indexed by a release number (e.g., 6.8.2) so a user can refer to a particular release and get reproducible results. A release will involve particular versions of packages. A package generally consists of a set of code for a particular task, together with a GNUmakefile, scripts and documentation. All BABAR software is maintained in AFS (Andrew File System) directories, so the code is accessible worldwide within the Collaboration. The combination SRT, CVS, AFS, has demonstrated to be a valid, powerful and efficient way of organizing the software infrastructure of a modern HEP experiment with collaborating Institutes distributed worldwide, both in a development and production phase.

1. INTRODUCTION

The BABAR experiment at SLAC (Stanford Linear Accelerator Center) is the ongoing effort involving more than 70 collaborating Institutes and Laboratories world wide. The resulting computing environment is therefore rather complex in terms of software management and distributed computing resources. BABAR has adopted an object-oriented approach for its software development with C++ as the principal programming language. Fortran and C are also supported.
Due to the large number of Collaborating Laboratories, multiple platform architectures and compilers are supported, mainly based on UNIX: SUN/SunOS, DEC/OSF, HP/HP-UX and in the future, also Linux. To support the software development, BABAR created a code management and release system [1,2] which allows integration of a distributed developer community.
The software infrastructure, the release mechanism and the underlining software architecture [3,4] are well integrated, offering an overall flexible and working system.

*Work supported by Department of Energy contract DE-AC03-76SF00515. Proceedings of the 6^{th} International Conference on Advanced Technology and Particle Physics, Como (Italy), October 1998

2. SRT - Software Release Tools

Since November 1994, BABAR has in place a system for code management and release control, SRT [1], which, based on CVS (Concurrent Versioning System [5]), consists of a set of GNUmake [6] scripts (tools) defining the BABAR release environment. Software releases can be made on different architectures and compilers in a homogeneous way, defining common rules used in the GNUmakefiles for controlling and executing the database setup, invoking in the right sequence all commands needed to compile and build:

- DB schemas;
- software libraries and binaries;
- documentation and test applications.

SRT provides several utilities to make easier managing the release process in a distributed environment; some of them include:

- *importver*, *importrel*: import a single package (specifying its version) or a whole release from a remote site;
- *updrel*, *rmrel*: update or remove a release specifying the release number.

- *auditrel*, *statusrel*: check for inconsistencies in a release or display its contents.
- *listtag*: display all tags of a CVS module.

2.1. Packages

The main BABAR CVS repository is located at SLAC and made accessible to collaborators worldwide via AFS [7]. BABAR software is organized into *packages*. SRT (*SoftRelTools*) is itself a package within the BABAR system. A package is a logical collection of classes/functions providing a set of services and performing a well defined task, i.e. what is defined to be a Booch *category*. A package usually contains:

- the source code;
- one GNUmakefile defining all rules specific to the package and complementing those already offered by SRT;
- documentations and notes;
- scripts and macros specific to the package.

Each package is maintained in the CVS repository as a separate module and is managed by one package coordinator. The package coordinator is responsible for the development of the package, for testing the code, and for tagging and announcing new self-consistent and tested versions of the package as and when appropriate for a general release.
Access to CVS modules can be controlled via AFS ACL's if required and package coordinators are notified automatically when changes in the code are made or when new code is checked in the repository for that package. Requests for creation of a new package must take place some time before the scheduled release by specifying: the name of the package, the initial tag, the name of the package coordinator and confirming the existence of a README file summarizing goals and functionalities of the package.

3. BaBar Software Releases

A "Software Release" in BABAR is a snapshot of a consistent set of package versions, including online and offline software, utilities, scripts and GNUmakefiles.
A Release is regularly built every two weeks and is indexed by a *release number* of the form: 6.9.X, where "X" is a number labeling a particular *build* of a release. There can be more than one build per release, usually happening in two days intervals, depending on how successful the final build is.
Periodically, some releases are labeled by name (by using symbolic links):

- **Test** Release: the final build of a release which has been considered successful;
- **Current** or **Production** Release: a Test Release which has been considered stable and therefore can be used for production;
- **Newest** Release: the latest Test Release available.

Usually, "test" and "newest" releases may have bugs which make them unsuited for general use. Releases pointed to by these links will change over time and it is also possible for two or all three of these links to point to the same release.
A release includes also libraries and binaries for each BABAR supported architecture/compiler. Currently, a release in BABAR consists of more than 300 libraries and 80 executables.
Figure 1. shows the BABAR software directory tree, which, based on AFS, has the actual CVS repository pointed by the variable $CVSROOT, while in $BFDIST are contained all the different packages exported by their tags (tags are used to represent the labels associated to their physical directories). $BFDIST also contains the list of all available releases by their release number, which in turn contains symbolic links to all package versions belonging to each particular release and all associated libraries and executables for each architecture; there's a 1-1 mapping between packages and libraries in SRT.
A relatively new and still evolving area is the online release structure. The "online_releases" directory has an organization similar to the (offline) "releases" directory. Online releases are designed to be dependent on a particular offline release. Some packages are common to both online and offline.

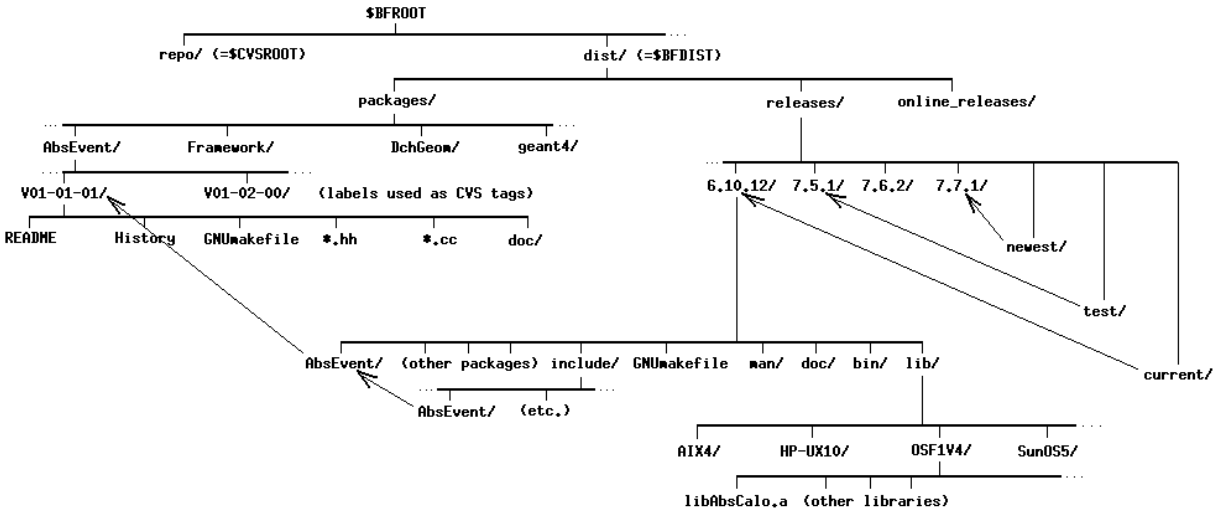

Figure 1. The BaBar Software Directory tree.

3.1. The working model

Package coordinators release their package(s) by committing, testing, tagging and announcing their code for a specified release.
A package's tag must have a fixed format, i.e. V01-25-08. Tag labels are also used to name the directories in $BFDIST containing the exported code for a particular version of a package. A tag can survive to more than one release until a new tag for that package is announced.
Package coordinators use an automated system based on Web, Hypernews and BFMAIL to announce their package version for a particular release, by filling a checklist Web form (see Fig.2).
There is a test compilation system available that compiles and tests on all platforms/compilers the announced package. The build will start automatically and the results will be posted by e-mail to the package coordinator.
All announcements are posted via BFMAIL to a distribution list and registered on Hypernews.
Nightly builds start automatically each night on all supported platforms, collecting all announced tags of the day. Problems occurring during the build are automatically posted to the package coordinators the day after.

Every two weeks, at the scheduled day and time all announced tags are automatically collected and the release build starts. If the release is not considered successful, it will be repeated at intervals of two days until the final release is announced by the release coordinator. A summary of each build is posted automatically to each package coordinator together with problems which may have occurred in the packages he is responsible for.

3.2. Automated testing

Package coordinators can provide specific tests and make them available at release time as regression tests. These tests (real applications) are then built and run automatically at each release build.
The output of the tests is automatically compared to reference output supplied by the developer.
The results of the regression test are then posted to the package coordinator for evaluation.

3.3. The BaBar Problem Tracking System

Errors from all release stages affecting a package are automatically documented and posted to the package coordinator. Each error or problem not fixed within a fixed period of time is automat-

Figure 2. The BaBar CheckList Web form.

ically appended to the BaBar *Problem Tracking System* which "gently" and regularly reminds every 7 days the package coordinator of the problem.
Outside release builds, bug reporting and tracking are controlled by this system. Users are free to submit problems ("open" problems) or "close" fixed problems.
The system, implemented over *Remedy ARWeb* and based on ORACLE, offers also the possibility to browse and print statistics of all tracked problems in the BaBar Software (Fig.3)

4. The user environment

The installation of SRT in the BaBar user environment is very simple and requires just a few

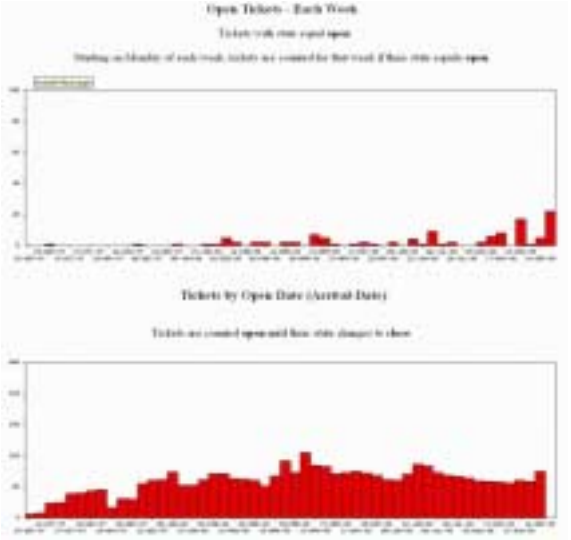

Figure 3. A snapshot of the BaBar Remedy Tracking System.

environment variable entries to be set in $PATH and $MANPATH. Centralized *HEPiX* scripts can take care of this.

4.1. The User Test release

A *user test release* consists of a "limited" collection of packages which are tied to a specific BaBar release. Users can modify packages or add their own new packages, and link them with the standard code and libraries which are part of a general BaBar release.
Common commands users ordinarily use are:

- **newrel**: to create the skeleton of a user test release tied to a specific release version;
- **addpkg**: to add an existing package in the test release (can also be a different version than the one available in the general release). *addpkg* deduces the tag needed for the package so that the developer needn't figure it out.

The skeleton of a test release usually includes:

- a *GNUmakefile*;

- *lib*, *bin* and *tmp* directories containing the products of the compilation;
- *shlib*, *shtmp* directories in case shared libraries are built;
- *doc* and *test* directories, where documentation and tests provided by the package coordinator for a given package are generated.

Only packages added to the test release are made part of the compilation. The final executables are built linking against those libraries in the test release and those (if needed) coming from the general release.

4.2. Tools: Workdir

Workdir is a simple package which allows the user to run applications built in SRT from any area, not necessarily within the user's test release. It consists of a "special" GNUmakefile integrated in the SRT environment, which gives the ability to easily choose in a flexible way which release (or test release) to refer to. It also creates links to the major scripts needed by applications at run time.

4.3. Tools: PackageList

PackageList is a package in SRT whose goal is to break the dependency of packages on their link list [8]. It defines a standard link order for all packages which are part of the BaBar software. It contains information pertaining to every SRT package, mainly:

1. the library name;
2. a file *link_$PACKAGE.mk* in each package listing all direct dependencies of that package on other packages.

Here is an example of the section associated to each package and placed in PackageList in a well defined order:

```
PACKAGELIST += L1Sim
  ifneq ($(LINK_L1Sim),)
  override LOADLIBES += -lL1Sim
  -include L1Sim/link_L1Sim.mk
endif
```

4.4. Tools for Quality Checking

Users can make use of specialized tools for quality checking and metrics monitoring. Usage of these tools (*Insure++*, *Code Wizard*, *Logiscope*, ...) is easily integrated in SRT.
Special release builds are performed regularly on "stable" or "production" releases, by filtering the code with these tools. Results of the filtering is published on the Web and the most relevant problems affecting a package are posted to each package coordinator.

5. The BaBar Application Framework

A *Framework* is a collection of classes that provide a set of services, generally providing a well defined interface for some particular task or set of tasks. The services provided by the BaBar *Application Framework* [9,10] (the *Framework* package) include input and output of event, calibration and control data, and management of the event processing loop. Changes in any of the i/o only affects the Framework and not the details of the BaBar reconstruction and simulation software. An obvious example of this separation is that the reconstruction software can be made independent of the external environment within which it is running, e.g. online or offline.
The basic unit of the Framework is the *module*. Modules contain code that takes various data from each event, runs specific algorithms, and puts the results back into the event for later use. The code comprising a module may do simulation, reconstruction or analysis tasks, and can either be common code from the BaBar release system or the user's private development in a test release.
Modules are generally independent of each other and have their own configuration and data-taking run specific information. Each module must provide an interface to the Framework which includes a unique name and functions to be called at the beginning/end of the job and event processing.
An executable program is built from one or more modules by compiling and linking them together. The individual modules may then be enabled or disabled at runtime and their order of execution specified through Tcl scripts.

Modules can be collected into *sequences*. A sequence of modules is a list of one or more modules that will be executed in the order by which they appear in the script.
Modules and Sequences can be combined into *Paths*. A Path is a complete execution sequence, commencing at the input module and terminating at the output module. The presence of *filter* modules within a path might terminate the execution of a path prematurely.
Coordinators of a Framework package may provide specialized tests to be built and run automatically at release time as regression-tests in a global context!

6. Hypernews and FAQ

In a distributed environment of a modern HEP experiment, the *World Wide Web* plays a fundamental rule as instrument for communication. Discussion Forums based on *Hypernews* and organized by topic are commonly adopted in BABAR. SRT and BFMAIL are integrated in this as well. Users are free to subscribe or unsubscribe to a particular topic and the addition of new topics can be easily managed.
A mechanism of *FAQ* (Frequently Asked Questions) is also in place, in order to summarize and focus on the most common problems developers and users may have.

7. Conclusions

BABAR has demonstrated that the SRT+CVS+Hypernews is a "working" combination for Software Management in the distributed environment of a modern HEP experiment.
SRT (Software Release Tools), originally developed in BABAR is now adopted and customized by more and more experiments world wide: FNAL, ATLAS, CMS, CDF, D0, BTeV. It requires an effort and a certain degree of responsibility by the user (the *Package Coordinator*) to achieve its best use. SRT offers a flexible and efficient user environment and can be easily extended with new tools. It fits well in the BABAR software infrastructure, together with an automated release mechanism, Hypernews and Web based tools and with the software architecture of the BABAR Application Framework.

REFERENCES

1. B. Jacobsen, The BABAR Software Release Structure, 1995.
2. J. Bartelt, User's Guide to the Software Release Tools, 1998.
3. B. Jacobsen and D. Johnson, Configuration Management, CSC98 - CERN School of Computing, Madeira, Portugal, 1998.
4. N. Geddes, The BABAR User Guide, 1998.
5. P. Cederqvist et al., Version Management with CVS, Signum Support AB, 1993.
6. R.M. Stallman and R. McGrath, GNU Make, Free Software Foundation, Boston, 1996.
7. The AFS User's Guide, Transarc Corporation.
8. B. Jacobsen, PackageList: The BABAR system for decoupling packages during linking, BABAR Note 443, 1998.
9. D.R. Quarrie, A BABAR Application Framework, 1995.
10. E. Frank et al., Architecture of the BABAR Reconstruction System, BABAR Note 357, 1997.