Astronomical Data Analysis Software and Systems XII
ASP Conference Series, Vol. 295, 2003
H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds.

CARMA Software Development

Marc W. Pound[1], N. S. Amarnath[1], Kevin P. Rauch[1], Peter J. Teuben[1], Stephen L. Scott[2], Rick Hobbs[2], Andrew D. Beard[2], Paul Daniel[2], David M. Mehringer[3], Raymond Plante[3], J. Colby Kraybill[4], Melvyn Wright[4], Erik Leitch[5]

Abstract. CARMA (Combined Array for Research in Millimeter-Wave Astronomy) will combine the existing BIMA and OVRO mm interferometers into a single array at a new high altitude site ($\sim$ 8000 ft). A third array, the Sunyaev-Zeldovich Array (SZA), will be built in the next two years and co-located with the CARMA interferometer. The SZA antennas will be available at times for cross-correlation with the CARMA antennas. This combination of heterogeneous antennas and their subsystems bring up new challenges not only in hardware, but also in software and in remote collaborations.

The two existing arrays have their own mature operations software, developed over the last decade, and the SZA software will be partially based on the DASI system currently at the South Pole. For CARMA, the situation is not as simple as choosing one over the other. It is further complicated by the fact that the software developers are dispersed among five institutions and three time zones. Such multi-institution development requires frequent communication, local oversight, and reliable code management tools.

Timeline has forced us to carefully balance reusing existing software, with perhaps wrappers to a new more object oriented approach, and rewriting from scratch. New hardware, such as the correlator, has already resulted in new software, but we anticipate re-using a fair fraction of the existing telescope software.

This paper summarizes our ideas on how we plan to do this, as well as outline what we call the CARMA Software Toolkit and associated Software Engineering aspects.

[1]University of Maryland

[2]California Institute of Technology/Owens Valley Radio Observatory

[3]NCSA/University of Illinois

[4]University of California, Berkeley

[5]University of Chicago

1. Introduction

Two preceding papers by Scott et al. (2003) and Plante et al. (2003) describe the COBRA/CARMA correlator and the need for a new CARMA native data format. The software required to control the CARMA array will be gradually phased in to co-exist with the current array control systems, which will be used as testbeds for the CARMA software.

BIMA uses one central computer, running Solaris (Hoffman et al. 1996; Welch et al. 1996; Yu 2001) controlling all antennas, and is currently starting to use Antenna Computers (AC) running Linux. Antenna Computers are housed in the antennas themselves and are intended to take over some of the functionality of the central computer (e.g., drive system control). OVRO has already been using AC's (microVax), with a VaxStation running VMS as the central control computer. Although the SZA is a newly built array, some of the existing software from the DASI system will be reused.

CARMA will be run by a central Array Control Computer (ACC), running Linux, and each antenna will have a diskless AC running real-time Linux. The ACs will be booted from flash ROM and use NFS to access system disks. All off-line software will be running Linux and Solaris, but MacOS X and Windows will most likely be used for monitoring purposes.

2. Software Reuse: Carma Software Toolkit

We identified four types of software components we will use for CARMA:

- **In-house** — the software we design ourselves, using CVS for version control, and for which we will have our own build system. We still expect some of the software in this category to be reusable from the original arrays.
- **Friendly** — written by colleagues or collaborating groups, and over which we have some control in terms of code development. Their build system is however not integrated into ours and we limit ourselves to using stable releases. Examples: MIRIAD, AIPS++.
- **Alien** — imported directly from other Open Source projects, but over which we have no direct control and whose build system will have to be understood by ours. Examples: pgplot, cfitsio, USNO ephemeris.
- **Commercial** — either in source code form, or binary only. Unlike the previous three, the distribution of this type will have to be controlled and cannot be open. Examples: ORBacus, IDL.

Together these components will make up the *Carma Software Toolkit* (CST).

3. Basic Tools

Source code management will be done using CVS. We will use CVS dead-end branches to create releases that will be used by observers to control the array. Feature enhancements, as well as coding experiments, are also encouraged to occur via branching, though meant to be merged back to the mainline development. Lastly, CVS has also proven to be a very useful backup tool.

C++ will be our main language for development, using the GNU compiler, though compilation using at least one other compiler and one other architecture is done to improve portability. The build system is configured using *autoconf* (a GNU toolkit), and a set of hierarchical makefiles. Documentation is extracted automatically from the source code via the *doxygen* tool.

4. Hardware Interfaces

On the hardware side, CARMA Interface Control Documents (ICDs) are intended to be the definitive documents governing the mechanical and electrical interfaces between various subsystems. Although some of the information will inevitably be included in documentation for the subsystems themselves, the ICDs are the governing document in the case of any discrepancies. The ICDs are stored in a linked system diagram, making it easy to traverse the CARMA system and see the interrelation of components, all the while having detailed documentation immediately available.

For software ICDs, we have followed a similar approach. Where a software system is strongly tied to hardware, we will create direct links from the hardware system diagram to the doxygen-generated high-level software API. For pure software ICDs, we will use a parallel package tree, also generated by doxygen. Documentation updates will be part of the nightly build.

Some subsystems have large autonomous software components: e.g., gpib, canbus, and cobra.

5. Software Engineering

We use a classical approach of writing requirements, design and implementation. Our work will be very distributed, across five locations and three timezones, aided by CVS for source code control. Nightly builds will keep track of the stability of the system. Due to the distributed development environment, off-site a number of hardware components need to be emulated. This will also simplify writing observing checkers. All workers meet during weekly tele-conferences and regular (twice a year) face-to-face meetings.

6. Work Packages

After analysis and prioritization of CARMA's high-level computing requirements, we have divided the work into 42 Work Packages, about half of which have a strong tie to hardware. Examples of work packages are Master Clock, Atmospheric Delay Correction, Monitoring, and Atomic Commands. Each package has been assigned to one or more institutions and a lead developer identified. The lead developer is principally responsible for design, implementation, and testing of the work package, as well as biweekly status reports to the CARMA Project Manager. In addition, each institution has a Work Package Manager (who is also one of the developers), whose job it is to keep an eye on local milestones and identify any scheduling problems that may arise.

References

Hoffman, W., Hudson, J., Sharpe, R. K., Grossman, A. W., Morgan, J. A., & Teuben, P. J. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 436.

Plante, R., Pound, M. W., Mehringer, D., Scott, S. L., Beard, A., Daniel, P., Hobbs, R., Kraybill, J. C., Wright, M., Leitch, E., Amarnath, N. S., Rauch, K. P., & Teuben, P. J. 2003, this volume, 269

Scott, S. L., Hobbs, R., Beard, A., Daniel, P., Mehringer, D., Plante, R., Kraybill, J. C., Wright, M., Leitch, E., Amarnath, N. S., Pound, M. W., Rauch, K. P., & Teuben, P. J. 2003, this volume, 265

Welch, W. J. et al. 1996, PASP, 108, 93

Yu, T. 2001, in ASP Conf. Ser., Vol. 238, Astronomical Data Analysis Software and Systems X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne (San Francisco: ASP), 495