The ALMA Computing Project – Update and Management Approach

B.E. Glendenning[1], G. Raffi[2]

[1]National Radio Astronomy Observatory, Socorro, New Mexico, USA

[2]European Southern Observatory, Garching, Germany

ABSTRACT

The Atacama Large Millimeter Array (ALMA) is a project to build a radio interferometric telescope containing a large number of antennas (nominally 64) at a high site in Chile operating in the mm and sub-mm spectral region. With the addition of a Japan in the last year, ALMA is now a global partnership with participation by institutes in Asia, Europe, and North America. The scope of the ALMA Software includes all aspects: observing script creation through GUIs, dynamic scheduling depending on weather and instrumental parameters, instrument control (including the correlator device, capable of producing data at more than 1 GB/s), data handling and formatting, data archiving and retrieval, and automatic and manual data processing systems. The scope has recently been increased to support telescope operations (e.g., referee support). This ambitious scope is being implemented by a very distributed team, with approximately 60 members at institutes in 10 countries. This paper will describe some technical highlights of the software system, some technical lessons learned after the initial deployments to support initial antenna prototype tests, and will describe some of the management approaches used to keep the software effort coherent across the entire project.

INTRODUCTION

ALMA is a large radio interferometer presently under construction at a high site in Chile. It is a global partnership of institutes from four continents. It is funded at the level of ~$750M (2000). The project was started as an equal partnership between Europe and North American institutes, and has more recently been joined by Japan. The funding organizations are:

- The US National Science Foundation (NSF)
- The National Research Council of Canada (NRC)
- The European Southern Observatory (ESO)
- Spain (MCYT-IGN)
- The National Astronomy Observatory of Japan (NAOJ)

ALMA is being built in cooperation with the Republic of Chile, which offers an exceptionally dry and transparent site (better than the South Pole for a substantial fraction of the time) in the Chajnantor area of the Atacama Desert. The ALMA project is currently undergoing a rebaselining process in reaction to cost growth.

ALMA consists of a set of 12m antennas (each pair an interferometer) capable of receiving in the frequency range 37 GHz to nearly 1 THz. While observing the antennas are on fixed pads, however they can be picked up and transported by large wheeled vehicles. Antenna separations can range up to nearly 14km, resulting in spatial resolutions as fine as 10 milli-arcseconds. NAOJ is providing four antennas optimized to provide total-power (i.e., autocorrelation) data as well as a compact array of 7m antennas to provide higher sensitivity to broad, low-surface brightness features (each interferometer is sensitive to angular size scales inversely proportional to their separation on the ground). There are a number of important scientific reasons to build ALMA, it perhaps suffices for this paper to note that most of the photons in the universe are emitted in the ALMA observing range.

The first production antenna will arrive on-site in late 2006, with an early science period starting in 2009, and the end of the Construction project at the end of 2011. As interim operations starts

ramping up in 2009, our construction staff will start ramping down. We assume in fact that the operations staff will largely consist of developers who transfer from the construction staff.

While the outcome is not yet certain, it is likely that the number of antennas in the main array will decline to 50 (from 64). The project is not recommending a decrease in the software budget, but until the process is finalized this cannot be stated with certainty.

SOFTWARE SCOPE

The fundamental data product of ALMA instrumentation is an irregularly sampled Fourier transform (with each pair of antennas at an instant returning a point (itself a multi-polarization spectrum) in Fourier space) of a portion of the sky, with various atmospheric and instrumental corruptions included. The desired output is a calibrated brightness distribution with two spatial axes, one spectral axis, and one, two, or four polarizations (usually one (Stokes I) or four (Stokes IQUV)).

It is the role of the ALMA Computing system to provide tools to support:

- The preparation and subsequent handling (e.g., refereeing) of observing proposals.
- To express the desired observing program for successful proposals, either in straightforward scientific parameters (target list, sensitivity, size scale, spectral and polarization parameters) or in very technical parameters for experts or for unusual projects.
- To dynamically schedule the observations so that the most challenging projects are observed with the best weather (relevant even at the outstanding Chilean site).
- To control the ALMA equipment (antennas, receivers, cryogenics, local oscillators, digital correlator, …) to carry out the observing program. Almost all equipment to be controlled is custom.
- To capture the output data from the digital correlator (at up to 1GB/s), apply certain online calibrations that have to be applied on a sub-integration timescale, time-average and reformat it and associate it with various ancillary data.
- To perform various online calibrations (e.g., to calibrate the antenna pointing on a ~10 minute interval).
- To provide operational controls, over-rides, alarms, health monitor, overview and detailed control panels, and quick-look displays of the raw and partially processed data.
- To archive all data (raw, observatory, instrumental) at a data rate of up to 60MB/s. (Average data rate in 2011 is specified to be 6MB/s = 180TB/y).
- To provide a processing “pipeline” to flag, calibrate, and image the raw data into reference images.
- To provide the Archival research access to the data, including through various “Virtual Observatory” initiatives which are developing cross-wavelength data access protocols and standards to promote data mining across observatory.
- To provide a desktop installable data reduction package for programs in which the pipeline images are not sufficient (for example, because data is observed in a non-standard mode).
- To provide support for various operational functions (e.g., data distribution to PI’s).

In addition to this functional scope, we have some additional funded activities which are not of direct interest to users, but are nevertheless required for the software development to proceed.

- Management.
- ALMA Common Software.
- High-level Analysis and design.
- Integration, test, and support.
- Scientific requirements support (testing, maintenance, expertise).
- Software Engineering.

Outside of our scope are pure business (MIS) functions, system and network administration, embedded programming "inside" devices (microprocessor, FPGA), and algorithm development.

RESOURCES AND ORGANIZATION

The worldwide construction budget for the ALMA Computing Integrated Product Team (IPT) is more than $45M (2000). This budget is dominated by personnel costs, including overhead and travel. It also includes about $6M for necessary operational equipment (computers, communications switches) and software licenses. Including in-kind personnel contributions it will be an over 400FTE-y effort. At present there are about 65FTE on staff distributed over 14 sites and 9 countries.

The IPT is jointly managed by the two authors of this paper, and K. Tatematsu (NAOJ). The IPT management jointly has complete technical authority over the project; in case of disagreement one of them (at present B.E. Glendenning, NRAO) is designated as the leader and is authorized to take technical decisions for the entire IPT. In practice we are able to arrive at decisions with consensus. Each manager is individually responsible for financial, contractual, and personnel issues in his own region.

At present the distribution of effort in the various areas is as follows:

Management, Process, Common Software

- Management 3 (excluding administrative support)
- Common Software (ACS) 8.6
- High level analysis, design 1.6
- Integration, test, support 4.8
- Scientific Software Requirements 2.9 (budgeted staff only, not counting in-kind contributions from committee members. Will be augmented in Japan).
- Software Engineering 2.5

Pre-observing

- Observing preparation/support 3.5

Observing

- ACA control 5.2
- Control software 7.7
- Correlator software 4
- Executive software 1
- Dynamic scheduling 1.1
- Telescope (online) calibration 2.9

Post-observing

- Archive 5.8 (really pre and post observing and operations!)
- Offline (AIPS++, simulation, data formats) 5.6 (ALMA construction budget)
- Pipeline (including quick look) 3.8 (budgeted, not counting in kind contributions from MPIfR, Paris)

Operations

- Observatory operations support software 0.5 (design phase)

MANAGEMENT APPROACH

The ALMA project has certain realities that must be accommodated in the management of its software development.

- It has an ambitious set of requirements.
- While the budget and team are very large for astronomy, it has been remarked (by our review panels, amongst others), they are nevertheless lean for the task (point 1).

- We have a team with very heterogeneous backgrounds (some with tremendous experience at observatories, others with none; some with considerable formal training, others with little). We are fortunate that the challenge of the project has attracted an exceptionally able development staff.
- We have a very distributed team.
- We have formal reporting requirements that tend to run counter to modern best practices ("agile").

Taking all of this, as well as "usual" software management issues, we have developed the following approach to managing the ALMA Software development.

1. *Requirements.* When the ALMA project was formed, while there was a rich heritage of science requirements, the requirements on the software system were much less developed. However it was understood certain features, like dynamic scheduling of observations, would be required of ALMA. We solved this problem by forming a committee or prominent mm band observers and staff with considerable experience with software systems at astronomical Observatories – the ALMA Science Software Requirements (SSR) committee. While their mandate was fundamentally to write requirements in areas of interest to potential ALMA Observers, it did include some operational requirements. The first product (2002) of this committee was a 160 page formal document containing numbered requirements and some use cases. While these requirements were enough to guide the overall outline of the system, they were not fine enough to track overall progress – most items would be in a "partially implemented" state for the lifetime of the development. Thus we have instituted a system in which the requirements needed for the next ~1 year of development are made more granular, by which we mean that they should become fine enough to be clearly delineated in the development and testing, but should not increase the scope of the subsystem. In addition to these Science requirements, a 60 page formal document on data processing requirements was produced also in 2002. This document is already granular enough to track development. A high-level set of operations requirements has been available since mid-2004, but they are still at a high level and need to be developed further, which is in progress.
2. *Architecture.* At an early planning stage of the project the work was divided into subsystems. After this a high-level architecture was defined to fulfil two purposes: to define the major business logic data flows between subsystems ("logical architecture"), and also to define some principles for how the software system should be organized to satisfy the technical constraints of the project (e.g., 200 computer system on 4 continents, 3 major languages (Java, C++, Python)). In retrospect the definition of the subsystems should have been deferred until after the architecture was available.
3. *Common Software.* An important strategy we adopted from the beginning was to require all developers to use a common software package, ACS, described elsewhere in these proceedings. The reason was to establish a common technical way of working in practice, not just an in principle result that would come from written standards. Amongst other advantages, it minimizes the support burden required of the maintenance staff who would otherwise have to absorb development differences from the very distributed and heterogeneous ALMA development team. ACS is basically the implementation of the ALMA technical architecture (for example, a multi-language container/component architecture based on CORBA) and a packaging of various services, some developed by ALMA and some adopted from elsewhere, for example an Alarm system from CERN.
4. *Oversight*. We have regular "contact" meetings with the subsystems, modelled loosely on progress meetings which would be held with an external contractor. These meetings concentrate on planned vs. actual development of features, action item status, test status, status of important bugs, and other issues hindering process. In addition to these meetings aimed at technical progress, each subsystem with a scientific or operational impact has a subsystem scientist appointed to oversee development with the view towards overall functionality and usability. The subsystem scientist also provides scientific input into the development process since the development team in many cases is unfamiliar with details of

radio astronomy. In addition, we have defined a series of reviews (one or more per year) to review both the overall state of the system and the design and other details for the coming year. In the case of software developed incrementally we do not consider it realistic to have a final design review with years of development still to occur. These reviews are sometimes internal, and sometimes external (defined by the top-level management), depending on the current state of development. To date we have held (and passed), an Internal Design Review, a Preliminary Design Review (PDR) (external), and three incremental Critical Design Reviews (CDR), one external. In addition we are subject to various other reviews called by (the project.

5. *Integrations*. In order to avoid the well known "integration hell" problem that would occur if the subsystems developed in isolation (with defined interfaces) before being finally delivered to a "big bang" integration in Chile, we have instituted a monthly integration cycle in which all subsystems are tagged and integrated together (running automated tests, looking for compile problems etc). To avoid the problem of interfaces between subsystems changing at the last moment before the monthly integration, we have instituted a policy in which the interfaces *between* subsystems (not within), are tagged with a two week offset to the monthly integration.
6. *Releases*. We have a release every 6 months, alternating major and minor. It is intended that the minor releases be interface compatible with the preceding major release when the project has sufficiently matured. Releases occur on a fixed schedule (in fact the releases use the monthly integration tag mechanism) rather than slipping the release date to maintain a fixed scope. In a widely distributed project it would otherwise be very difficult to otherwise synchronize delivery from a number of subsystems, and we consider it useful to have an understood development pace to which everyone adheres. Additional functionality can always be added via a patch if necessary.
7. *Planning*. We have a high-level roadmap, tied to the overall commissioning schedule, or which functionality is required through the end of the construction project. This roadmap is too coarse for detailed planning. For detailed planning once per year each subsystem is requested to provide a set of features they will develop in the coming year, and this is reviewed at the yearly incremental CDR. The expected completion date, testing method, and estimated effort are also recorded. This then naturally also forms the basis of evaluating the progress of the subsystem. Features are usually just a granular requirement for a subsystem, but also include a technical items required by the system but not visible to end users. The planned features, their status, and their relationship with the high level requirements are tied together in a commercial database (the Telelogic DOORS application).
8. *Testing*. Subsystems provide unit tests which are automatically executed every night. Failures are automatically distributed to individuals responsible for a particular module. For subsystems with visible scientific or operational concerns external user tests (with members solicited from the community) are arranged to test the subsystem *in isolation* by representative users of that subsystem. This usually results in a written report, a questionnaire from each tester, and an evaluation ("grade") for each evaluated requirement. The test is aimed at the requirements fulfilled in the previous 6-month development cycle. The integrations result in technical tests and procedures which can be run in a regression sense. In 2006 we will start a campaign of integrated user tests, in which the integrated software system will be used at our 2-antenna test facility in New Mexico. We will do these tests organized by observing mode in the order they are needed in Chile. This will result in regression tests, commissioning scripts, and evaluation of requirements that can only be tested in an integrated system.
9. *Communications*. ALMA is a very distributed project, and software development requires a lot of discussion and collaboration. We have adopted the following approaches. Subsystems with team members in multiple locations typically have a (typically) weekly phone meeting (video meetings in general have not been reliable for, owing to disparities in network bandwidth and equipment availability between sites). We have a significant travel budget, budgeted at the level of 3 trips/developer/year. We use various electronic communications

tools (Yahoo messenger and Skype are popular). And for collaborative written discussions the Wiki (TWiki version) has been enormously effective for us.

While we evolved the above system to meet our own particular needs, it is always interesting to compare one's own practices against those of others. Comparing the ALMA system against the 6 core practices of the Rational Unified Process (as summarized in the very nice survey of modern agile software development practices, [1]).

1. *Timeboxed iterations of 2-6 weeks*. Our monthly integrations provide this.
2. *Cohesive architecture, strive to reuse existing components*. We have a cohesive logical and technical architecture (ACS). ACS in turn is based on software which already exists, although ALMA has considerably enhanced it.
3. *Continuously verify quality, test early, integrate and test*. In general we are putting appropriate effort into integration and integrated tests and standalone user tests. We collect various software quality metrics, but they are not paid sufficient notice. Our unit tests are not sufficient for all subsystems.
4. *Visual modelling before iteration.* Our fundamental data models are developed in UML and code (language bindings, XML schemas) are then automatically generated from the UML. We do not otherwise insist on any use of a visual modelling language, which people then use as they find appropriate for their subsystem. Many do not use them at all.
5. *Manage requirements*. We have expended considerable effort in this area.
6. *Manage change.* All software and related files (e.g., makefiles) are kept in a version control system (CVS). Intervals in which inter-subsystem interfaces can change are prescribed. We have a software change control board (SCCB), but it does not yet consider most changes to the software. As the project moves closer to completion we will increasingly place items under the control of the SCCB.

ACKNOWLEDGEMENTS

The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc

REFERENCES

[1] *Agile and Iterative Development: A Managers Guide*, Craig Larman, Addison-Wesley Professional (2003).