ALMA Software Project Management Lessons Learned

G. Raffi[1 (*)], B.E. Glendenning[2]
[1]European Southern Observatory, Garching, Germany
[2]National Radio Astronomy Observatory, Socorro, New Mexico, USA

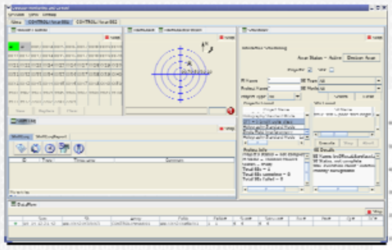
ALMA High Site (5000m)

Abstract

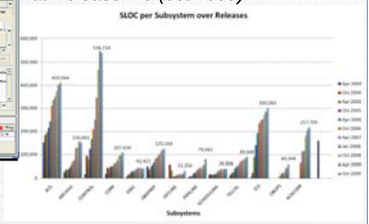
The Atacama Large Millimeter/Submillimeter Array (ALMA) is the largest radio telescope currently under construction by a world-wide collaboration. The first antennas (the total will be 54 12m antennas and 12 7m antennas) are being commissioned to become part of the interferometer at a high site (5000m) in Chile. The ALMA Software (~ 70% completed) is in daily use and was developed as an end-to-end system including proposal preparation, dynamic scheduling, instrument control, data archiving, automatic and manual data processing, and support for operations. The management lessons learned will be explained. Aspects described will go from requirements analysis to the use of a development framework: ALMA Common Software (ACS) in our case. The process used to provide regular releases will be outlined, including temporary cross-subsystem teams. The importance of integrated regression tests will be stressed, but also the need to validate the system with users. Among the project management tools risk analysis, earned value measures and tracking of requirements completion will be presented. Monitoring progress with reviews and the possible impact on completion dates will also be discussed.

ALMA Software Project Highlights

- ➢ALMA software is an end-to-end system:
  - ▪Proposal and observing preparation
  - ▪Dynamic scheduling
  - ▪Instrument control
  - ▪Data archiving
  - ▪Automatic and manual data processing
  - ▪Support for operations
- ➢Developers are distributed:
  - ▪Over 4 continents and 15 locations.
  - ▪About 80 people on software development and testing
  - ▪Size over 2000 kLines of specially developed source code (so far). >75% complete.
  - ▪In use for commissioning of ALMA observatory


Operator User Interface

Lines of code by Computing subsystem at Release 7.0 (oct.2009)

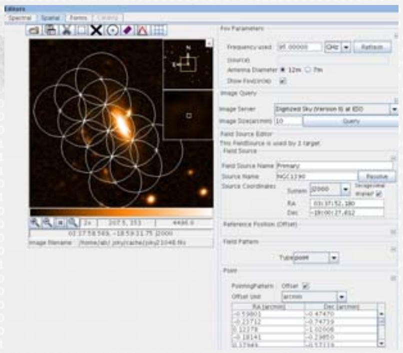


The ALMA software is in regular use at the ALMA Observatory since more than a year (having been previously tested on prototype antennas) **and keeps being incrementally updated at every new Release.**

Project Management Aspects

- ❖Distributed Team Management (Most important: **Wiki**, CVS, regular telecons, face to **face** meetings)
- ❖Requirement and tracking progress
- ❖Use of a development framework: **ALMA Common Software (ACS) (#)**
  - ➢Software written with ACS implicitly uses its architecture. This is good for distributed development to maintain consistency between different developers.
- ❖Releases at fixed dates and Planning
- ❖Temporary cross-subsystem teams
- ❖Integrated regression tests by an independent team
  - ➢And user tests (both stand-alone and on integrated system)
- ❖Project management tools: risk analysis, earned value measures
- ❖Reviews to monitor progress: Internal and External
- ❖Problem reporting (JIRA system)

Observation Preparation: Spatial Editor (used to prepare a mosaic of observations – UK ATC/ESO)



(#) Based on the container-component paradigm and using CORBA. The system allows the use of C++, JAVA and Python on Linux operating systems.

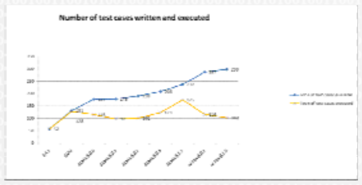
For more information on the ALMA Common Software see poster and paper:

–***TUP101*** *ALMA Common Software (ACS), status and development by G. Chiozzi (ESO) et al.*
–***WEA006*** *Data Distribution Service as an alternative to CORBA Notify Service for the ALMA Common Software by G. Chiozzi (ESO) et al.*

Lessons Learned

- ❖Requirements collection (with Use Cases) was important
  - ❖There will be still missing or late requirements, but design is done upfront
  - ➢Requirements working group to be recommended
  - ➢Tracking requirements completion to show progress (planned vs. actual)
- ❖Using a software framework is essential (ACS) (but most of this would apply also to EPICS, TANGO etc):
  - ❖Allows collaborative work, results in an homogeneous system
  - ❖Provides a solid debugged base of software
  - ❖Enforces also hardware standards and operating system versions
  - ❖Makes large distributed projects manageable and maintainable
  - ➢.. But requires team discipline and managerial support
  - ➢And learning (yearly ACS courses in our case)
- ❖Incremental Releases at fixed dates (vs. fixed content) twice/year
  - ❖Software is developed incrementally in 6 monthly steps
  - ❖Easier integration, predictable dates for the rest of the project
  - ❖Releases are an integrated e2e system
  - ❖Patches (typically one per 6 month development cycle) allow to upgrade a few computing subsystems
  - ❖Planning work is for 6 months and can be tuned to accommodate project priorities
  - ➢Give priority to testing and making releases over development when deadlines approach
- ❖Cross-subsystem Function Based Teams (FBTs) (~3 months)
  - ❖Implement important functionality reducing impact of changing interfaces
  - ❖Make integration easier, as inter-subsystem issues get sorted out continuously.
  - ❖Integrations are more frequent, which is important with a geographically distributed team
- ❖Integration tests (by independent team)
  - ❖In addition to subsystem tests (build-in test time up front)
  - ❖Regression tests, eventually mostly automatic
  - ➢Require good test models (several computers)
  - ➢… but cannot replace tests with real hardware
  - ➢defend towards the rest of the project the need for significant test time on the system, to discover/fix issues before software gets used
  - ➢.. You will get anyway criticism later and it will be your problem if you did not follow your procedures
- ❖Problem reporting (JIRA in our case)
  - ❖Important to track bugs/improvement request
  - ➢JIRA is good. Whatever the system, follow up is even more important
  - ✓ We have a weekly meeting to discuss issues and flag blocking ones
  - ✓Have a fallback release available for use, in case of major problems
- ❖Project management tools:
  - ✓Risk analysis helps project to assess software risks
  - ➢Earned Value (apart from Requirements tracking) was difficult for us to apply in a meaningful way
- ❖Reviews to monitor progress:
  - ✓Internal reviews , like Releases, are incremental. This allows incremental design and adjustment of priorities
  - ✓External reviews are good. They require preparation and thinking and result in obtaining comments that help in the remaining work.

ALMA antenna moving up to the high site (5000m). ALMA base camp (3000m) in the background.



Tests (shown here) are done on computer test environments. Final tests at the Observatory.

(*) graffi@eso.org

Acknowledgments

As the managers of the bilateral part of the ALMA Computing team we are proud to acknowledge the dedication of the ALMA Computing staff at ESO, NAOJ, NRAO and other Institutes around the world, who are doing a great job developing and supporting the software for ALMA.

Atacama Large Millimeter/Submillimeter Array