ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA014

EVOLUTION IN THE DEVELOPMENT OF THE ITALIAN SINGLE-DISH CONTROL SYSTEM (DISCOS)

A. Orlati, M. Bartolini, S. Righini, INAF - IRA, Bologna, Italy
M. Buttu, A. Fara, C. Migoni, S. Poppi, INAF - OAC, Selargius (CA), Italy

Abstract

DISCOS [1] is a control system developed by the Italian National Institute for Astrophysics (INAF) and currently in use at three radio telescope facilities of Medicina, Noto and the Sardinia Radio Telescope (SRT) [2]. DISCOS development is based on the adoption of the ALMA Common Software (ACS) [3] framework. During the last two years, besides assisting the astronomical commissioning of the newly-built SRT and enabling its early science program, the control system has undergone some major upgrades. The long-awaited transition to a recent ACS version was performed, migrating the whole code base to 64 bit operative system and compilers, addressing the obsolescence problem that was causing a major technical debt to the project. This opportunity allowed us to perform some refactoring, in order to implement improved logging and resource management. During this transition the code management platform was migrated to a git-based versioning system and the continuous integration platform was modified to accommodate these changes. Further upgrades included the system completion at Noto and the expansion to handle new digital backends.

INTRODUCTION

The SRT just went through 5 months of no-operations during which a complete refurbishment of the Active Surface System (SSA), including the repainting of primary mirror panels, was completed. In parallel the Data Center (included the control system workstations) has been moved from the original location to a shielded room. Exploiting this idle period the DISCOS staff, working at the SRT, heavily focused on the porting of our codebase to a newer ACS version and to put in place an even better automatic provisioning system in order to improve the quality of our testing process and the stability of our products. Similarly, in the pursuit of creating a good testing environment, some simulators of the telescope devices have been developed: currently the Active Surface, the Antenna Control Unit and Minor Servo System Unit can be simulated and the core of the DISCOS control software tests can be executed without accessing the telescope's hardware. This idle period was also exploited by the project's staff to update the production system on Medicina and Noto telescopes.

In the following chapters the major activities and areas of upgrade related to the DISCOS framework are described in greater detail. A whole chapter is dedicated to core upgrades in the control system. Another major activity has been the upgrade of the IT infrastructures supporting code development and code documentation. The two final chapters are dedicated to the description of how Medicina and Noto radio telescopes could benefit from the development done for the SRT and finally got upgraded version of the control system installed on site, enabling better performance and further capabilities.

DISCOS

The control system has gone through some maintenance and some major upgrades in this period of time.

Upgrade of ACS Version

As the previous development of DISCOS was based on a legacy version of the Alma Common Software framework this was causing major issues, mainly regarding the obsolescence of the whole system. Being tied to an old product not only means being unable to incorporate new ACS features and fixes, but the consequences are spread all over the control environment: the main one being the inability to upgrade operative systems to recent versions and the consequent maintenance of old products.

This was causing security issues and was also slowing doen the adoption of modern libraries and tools in the development of all the accessory software outside of the control system.

The main issues encountered in the porting activity are related to the different version of C++ compiler, c libraries and external libraries such as boost. Several unresolved symbol in the linking phase required to be addressed to the point that our make files are now completed changed. On the IDL side, we encountered some naming clash between types defined in our code base and types defined in the new versions of the language, but this required minor changes.

We were expecting troubles related to the migration to a 64 bit architecture, but none of those emerged so far.

Refactoring

The upgrade to a modern OS and the related activity was the first step for a full review of the code base. A large portion of the code based in fact was not exploiting recent additions made to the control system libraries for the standardization of logging behaviours and internal resource handling. This is being incorporated as part of the porting activity and is reducing the percentage of duplicated code, easing future maintenance.

As code is being ported to the new platform, and thus completely reviewed just to eliminate compile warnings

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA014

and fixing compile issues, log messages are being refactored according to new logging guidelines which better specify the format of log messages and how to assign logging levels. A differentiation between system logs and user logs has been introduced making it clearer what messages will be displayed to the operators and what messages will be only collected in system log files.

Another bigger refactoring undergoing the porting activity is related to the reference of components in the ACS component/container model. One of the major sources of duplicated code in the DISCOS code base is in fact related to referencing components from within other components. This is obviously using ACS and CORBA internals and can result in failures in finding components, or failures instantiating components, or in the need to reallocate a dead reference to a no-more-existing component. It is easy to understand how such activity can lead to runtime errors, referencing to non-existing objects or failing to release resources. For this purpose, a new template library has been developed which standardizes component referencing and eliminates a lot of duplicated code, also better adopting smart pointers thus easing resource deallocation.

Refactoring also interest the defect removal during the porting process. During this activity a new process has been introduced for defect removal and bug fixing where developers are using a Test Driven Development (TDD) approach, writing a test that reproduces the incorrect behaviour before fixing it. This enable regression tests to be effective.

New Features

Finalizing the scientific commissioning of the SRT involved the implementation of some features as part of the control system. In this section we give a brief summary of most recent and relevant additions to the DISCOS framework.

External Backends: A new protocol has been implemented for the integration of externally managed digital backends into the control system. The definition of the protocol consists of a request-reply communication on top of a TCP/IP connection. A C++ library has been developed implementing the protocol on the client side (the control system) while a reference protocol implementation based on Python Twisted is provided as a skeleton implementation for backend integration.

Roach based spectrometer: A new spectrometer based on ROACH2 hardware is being used in observations at SRT and Medicina sites. Integration with the control system is based on the aforementioned protocol.

Fits: Data format has been slightly modified in order to take into account the on-source or off-source position of the target with respect to the observation. Also a finalizing software tool is being integrated with the control system enabling the integration with a remote data archive.

Dewar positioner: The DewarPositioner component is in charge of compensating the field rotation when dealing with multi-pixel receivers (if equipped with a derotation system).

Frequency Tracking: In order to fulfill the requirements for spectroscopy observations, the system can now keep a requested rest frequency in centre of the band of the backend (section). Both frontend and backend local oscillator could be tuned during this operation. This capability is now supported from both command line and schedule.

Hardware Simulators: A simulator project has been bootstrapped, collecting hardware simulators of different devices under the same umbrella. The projects implements a micro-framework for simulator development, at the moment comprising: ACU simulator, Active Surface simulator, IF distributor simulator.

SUPPORT INFRASTUCTURES

The upgrade activities were made possible via an update of many support structures for code development, code hosting and provisioning of IT environments.

GIT Version Control

DISCOS project development has been using a private, self-hosted, subversion repository in the latest years.

As the project grows with different product lines and different tags and versions of the configuration database, subversion suffers of performance issues, up to the point where it becomes unpractical and difficult to manage. Primary for this reason the team started to investigate the possibility to migrate the entire codebase towards a git platform.

A github organisation [4] was created for this purpose, named after the project itself. The first project to be migrated to this platform has been the whole documentation, which has been reformatted to Restructured Text (RST) in order to be published automatically via the readthedocs online service [5]. The process of documentation translation to RST and migration to the github platform enabled everyone in the development team to practice with git distributed model and git development practices such as pull requests.

After the successful experiment of migrating the documentation it was decided and agreed that git would represent the best solution also for the actual codebase.

Migrating from subversion retaining the whole history of commits has been possible tying subversion user names to email addresses used by git and github users. Every subversion tag has been retained as git tag while the trunk development branch has turned into git master. Short lived branches are now used for the development of new features and the resolution of software defects. Subversion branches have been mapped one to one into git branches.

One of the biggest advantages using git is that a local copy of the repository is now occupying more or less 130MB of disk space while the complete subversion repository was using more than 3GB of disk space. This enables faster searches and meta data queries.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA014

The issue tracking system has also been ported to github, enabling a tighter integration of issues with the code base and their resolution based on pull-requests mechanism. The previous issues collected in a Mantis based system have been exported to an excel spreadsheet and automatically inserted into github using publicly available rest APIs.

Ansible Provisioning

In order to enable every developer to work on the porting of the code base it was necessary to streamline and standardize the process of creating the new development environment.

Based on the lessons learned implementing the AZDORA [4] continuous integration platform, an automated approach to environment creation and provisioning was adopted: instead of using bash scripts, this time the Ansbile automation tool was used to define tasks and resources to be configured. The basic advantage of using Ansible tool is the major configurability and built-in capabilities of the tool enabling a more coherent personalization of resources based on the telescope site. Also, the provisioning code can be better maintained in a single coherent space, and can evolve together with the project itself.

The final toolchain is in fact using Ansible to provision the environment of a development box created on a VirtualBox virtual machine managed using the Vagrant tool.

A simple automation has been developed so that the developer can easily switch development branch changing every system configuration involved.

In addition to the development configuration a production configuration is being developed using Ansible tools, trying to take into account the real deployment topology of machines and networks at the different sites.

This approach has many advantages as the reproduction of development and production environments can be consistent exploiting the same mechanisms, thus enhancing test reliability and defects reproducibility.

Capturing environment provisioning and configuration as code has also the advantage of automatically documenting sysadmin tasks that are often dependent on particular expertise or particular people thus becoming a bottleneck in the deployment process.

The obvious downside of this approach is that developers and sysadmins need discipline in the adoption of the tools for any change they need to perform on the development or production system. A starting phase of experimentation on the development side is undergoing, still pending the adoption in production environment.

NOTO

In early 2015 a prototypal version of the DISCOS control system was installed at Noto radio telescope. This version showed signs of poor stability resulting in difficult adoption by local operators who were continually facing issues with the control system.

A week of on-site development concentrated in late November 2016 was effective in incorporating the major changes from the Common code base into the Noto installation and resolved many open issues in the system interfacing local hardware. The new system now replaces an old PC used for antenna pointing that was causing obsolescence issues, being impossible to replace with the same hardware.

This upgraded Noto installation of the DISCOS framework is used for the characterization of the Noto radio telescope during the installation of new receivers and during its maintenance. The main differences between Noto product line and the other two installations is that many interfaces are not integrated into the control system which is composed only by the strictly necessary modules while delegating many functionalities to external components. External interfaces include: control of receivers via HPIB interface, control of the Active Surface and of the Sub-reflector via a TCP/IP based protocol. Other activities carried on during the busy week of development included: interfacing with local Total Power backend enabling programmatic noise calibration mark activation, adapting the weather station component to local hardware, automating boot procedures and setup of the local operator environment, installing operator tools for the generation of schedules and monitoring of observations, updating the developer and user's documentation, performing test observations and end-to-end system testing.

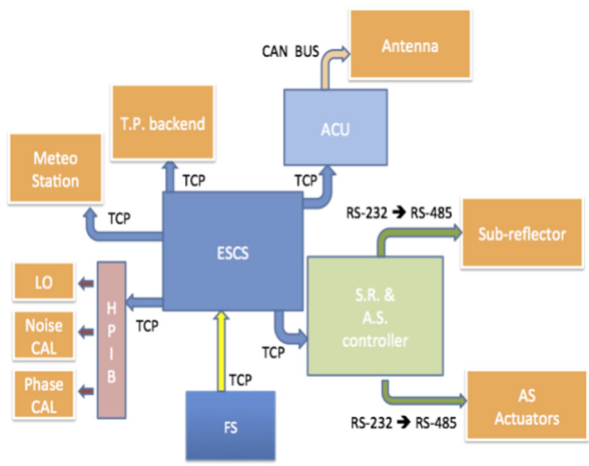

Figure 2 Block diagram describing DISCOS installation (ESCS) and customization for Noto radio telescope

MEDICINA

The activity related to the Sardinia Radio Telescope has been the primary commitment of the DISCOS team for a long period and this led to fewer upgrades being ported in the Medicina control system that was not upgraded in production in the last two years.

The end of the SRT commissioning left some bandwidth available to upgrade the system on the Medicina radio telescope, leading to the release of discos-med-0.6 in March 2017. Due to the modular nature of the control system, the whole Common code base shared by every telescope has been successfully implemented at Medicina site without major issues.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA014

A major upgrade at the Medicina radio telescope regarded the adoption of a Roach2 based spectrometer, equipped with the same processing firmware used at the SRT site. The adoption of such a backend has triggered some necessary changes in the data recording technology present at Medicina station: a new NAS storage has been installed and configured, equipped with 10gbps NICs and a hardware RAID controller, enabling the recording of fast data dumps by the spectrometer which can reach time resolutions up to one spectral data each 1ms.

The storage implements a Lustre filesystem which is also exported as a NFS filesystem for compatibility reasons with the old operative systems of parts of the control system. This is one of the many reasons for which a major DISCOS upgrade is necessary in order to address the obsolescence issue.

CONCLUSION

In the last two years DISCOS-based control systems enabled a full range of activities at Italian radio telescopes, enabling the commissioning of the Sardinia Radio Telescope, and an increased utilization of Medicina and Noto radio telescopes for single dish activities. Despite its stability, the system is undergoing some major changes due to address the obsolescence issue represented by an old version of the ACS framework adopted.

This upgrade process also represents the chance to adopt modern practices in code development management, migrating the system to modern distributed version control systems, migrating the provisioning technology towards a fully automated approach enhancing future maintainability and consistency of the software products delivered.

REFERENCES

[1] A. Orlati, *et al.*, "Design Strategies in the Development of the Italian Single-dish Control System", ICALEPCS2015, Melbourne, Australia, 2015.

[2] P. Bolli, *et al.*, "SRT: General Description, Technical Commissioning and First Light", https://arxiv.org/abs/1603.06134

[3] G. Chiozzi, *et al.*, "The ALMA Common Software: a developer-friendly CORBA-based framework", Proc. SPIE vol 6274, September 2004, p. 205 (2004).

[4] DISCOS github organisation, https://github.com/discos

[5] DISCOS online documentation, https://dicsos.readthedocs.io