ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

DISCOS UPDATES

S. Poppi*, M. Buttu, G.Carboni, A.Fara, C. Migoni INAF - OA Cagliari, [09047] Selargius, Italy
M. De Biaggi, A. Orlati, S. Righini, F. R. Vitello, INAF - IRA, [40138] Bologna, Italy
M.Landoni[1], INAF - Osseravatorio Astronomico di Brera, Merate, Italy
[1]also at INAF - OA Cagliari

Abstract

DISCOS is the control software of the Italian Radio Telescopes and it is based on the Alma Control Software. The project core started during the construction of the Sardinia Radio Telescope (SRT) and it further developed to support also the other antennas managed by INAF (National Institute for Astrophysics), which are the Noto and the Medicina antennas. Not only DISCOS controls all the telescope subsystems - like servo systems, backends, receivers and active optic system - but it also allows users to exploit a variety of observing strategies. In addition, many tools and high-level applications for observers have been produced over time. The development of this software follows test-driven methodologies, which, together with real hardware simulation and automated deployment, speed up testing and maintenance. We here describe the status of the DISCOS project and of the related activities, also presenting its ongoing upgrades.

INTRODUCTION

The Italian National Institute for Astrophysics (INAF) manages three single-dish radio telescopes: The Sardinia Radio Telescope (SRT), the Noto and the Medicina radio telescopes. These are open sky facilities; the international scientific community is invited to submit observing projects through calls for proposal, published twice a year [1]. The telescopes cover radio bands from 305 MHz up to 26.5 GHz, allowing many research topics to be explored. Examples are pulsars, astrochemistry, extragalactic sources, space weather. SRT and Noto are already provided with an active surface, allowing for observations at much higher frequencies; Medicina is planned to have it installed within spring 2023.

The control software plays a key role in an observing facility, allowing the users to perform the needed observations by using proper strategies and modes, while ensuring the quality of the acquired data. Therefore, in 2004 we started developing NURAGHE, the SRT control software. In 2007 we parallely began the ESCS (Enhanced Single-dish Control Software) project, devoted to the Medicina and Noto radio telescopes. Eventually, in order to optimize the efforts, in 2015 the three development lines were unified in DISCOS, a common control software for all the three telescopes.

DISCOS is built on top of the Alma Common Software, which is based on CORBA [2]. This framework allowed us to realize a modular software mostly made of common codebase, reused and deployed at all sites, as much as possible. Considering this, only a small part of the codebase (23%) is telescope-specific, essentially in the low-level and no-logic control of the devices and of the telescope hardware [3]. In 2017 we refactored part of the code, in order to adapt it to the upgrade of the framework to a newer version of ACS [4]. Also, we chose Github [5] to track issues and manage version control.

Equally important is the development strategy. We followed guidelines in the adoption of an approach called Behavior Driven Development (BDD) [6] which aims to test the software behaviour and it is used together with Test Driven Development (TDD) and unit testing strategies [7].

In the following sections we show the ongoing and planned DISCOS upgrades. In particular we present the integration of new instrumentation, such as new receivers and backends, a new simulated environment of the SRT hardware devices, a middleware DISCOS wrapper called SURICATE and a simple web-based monitoring and alarm system.

TELESCOPES UPGRADE

In 2019 INAF was granted a PON (National Operational Program) funding to upgrade the Sardinia Radio Telescope and its infrastructure toward higher frequencies (up to 100 GHz). Within this scope, up to a 15% of the overall funding was aimed to also upgrade the Medicina and Noto radio telescopes. The funded project includes the acquisition and the installation of new receivers on the telescopes:

- Three coaxials receivers K/Q/W band (18–26 GHz, 34–50 GHz, 80–116 GHz). for Medicina, Noto and SRT.
- 16-Beam W Band receiver (70–116 GHz) for SRT
- 19-Beam Q Band receiver (33–50 GHz) for SRT
- A bolometric millimetre camera for SRT operating in the 77–103 GHz made of 408 detectors.

Furthermore, the PON project includes, as concerns the SRT, the procurement of three digital data acquisition systems (backends), a new state-of-the-art metrology system, an HPC system and laboratory instrumentation. Also, the telescope minor servo system, which is responsible for the proper positioning of the telescope optics, will undergo a refactoring and a major upgrade.

It is worth noting that new receivers and new backends will need a big effort in order to integrate them in the control software. To do this, we exploited the ACS architecture and the DISCOS modularity. Not only the new receivers have the same interfaces, but also the communication protocols are the same of the SRT first-light receivers.

* sergio.poppi@inaf.it

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

The availability of new instrumentation will allow users to exploit additional observing modes, so we are also upgrading BASIE (the schedule creator), which is a fundamental tool to plan and execute the observations. BASIE was designed to create the schedule files required by DISCOS to carry out continuum and spectroscopy observations, according to different strategies - like on-the-fly scanning, on-off and nodding [8]. Thanks to this tool, users do not need to deal with the writing of the complex schedule files: they are only asked to specify a combination of receiver and backend, providing basic information on the celestial sources to be observed, the desired strategy and the configuration of the devices.

SIMULATORS

Writing a simulator helps the developers in writing more reliable code for the actual control software of the radio telescope, the DISCOS control software. Being able to test the code without having to rely on the hardware represents a huge advantage in the development and maintenance processes. It speeds up the development of new features and the bug fixing, keeping under control the codebase through the adoption of continuous integration practices. Both unit and functional tests are cleaner and shorter, no code mocking is required, they work both over simulators and real hardware. In addition, the integration of new components is easier and more reliable, and the hardware itself can be verified by running the simulator tests over it. A suite of simulators, capable of reproducing different scenarios, can be exploited to write and execute a great variety of tests whenever a modification to the control software code gets pushed to the main repository. All these considerations led us to change our process to integrate new devices, so that the writing of its simulator is now the mandatory first step. To improve and speed up our capability to add simulators to our environment, we developed a simulator framework. The framework for the simulators in written in Python and is composed of two layers. The topmost layer is in charge of handling network communications, it behaves as a server, listening for incoming connections from clients and relaying every received byte to the other layer of the framework, the simulation layer. This layer is where received packets are parsed and executed. It can faithfully replicate the behavior of the hardware, or it can simply act as a protocol interpreter by providing back proper answers. [9]

Furthermore, the framework allows to test how the control software code reacts under expected error conditions. In fact, it provides an easy way to simulate unlikely scenarios that are very difficult or, in some cases, impossible to replicate by only using the hardware. This allows the developers to write more reliable and robust code, likely permitting the recovery from an error condition without having to resort to a complete reboot of the system.

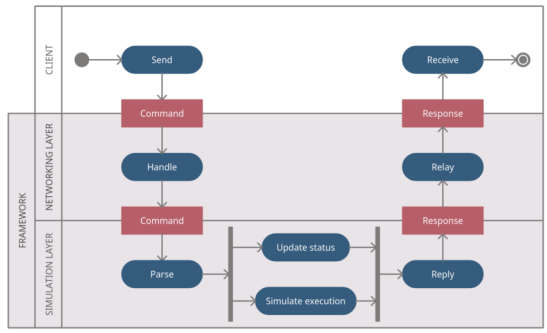

Figure 1: Simulators framework layers and communication behavior.

DEPLOYMENT

An automatic Ansible [10] pipeline creates users, configures the network, sets up the shared file system, installs all the required dependencies (i.e.: Alma Common Software), installs all the required observing tools and finally the DISCOS software, creating a fully-configured working environment. Having an automated deployment means that both development and production environments are aligned, all the process is automatically documented and the system can be easily updated and restored. Not only Ansible allows the deployment on physical machines, but it can also provide, through Vagrant, virtual machines based on Oracle VirtualBox.

A DISCOS installation is basically composed of:

- A manager hosting ACS services and maintenance tools
- A console, providing the user interfaces and accounts for all the allowed observing projects
- Storage cluster, mounted with LustreFS

The deployment scripts allow to select which telescope the installation is specific for and which branch or the tag from the Github repository is to be provided [11].

In summary, all the DISCOS instances are well aligned among all the installations. This allows us to speed up maintenance and bug tracking.

SURICATE

Suricate [12] is a middleware which exposes APIs to the clients and offers an abstraction from the control system and the programming language. Moreover, it allows DISCOS to be easily extended bypassing the framework on which it is based, getting advantage of new technologies.

It is composed of a sampler that collects from DISCOS the telescope status, its configuration and relevant parameters, writing them in on a in-memory database (redis-db). Then a different module, the db-filler, gets the data from the redis-db to save them into a persistent SQL database. A HTTP server gets requests from clients and reads the values from the database, returning a JSON answer. Also, the HTTP server

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-FRAL02

receives user commands from clients, forwarding them to DISCOS.

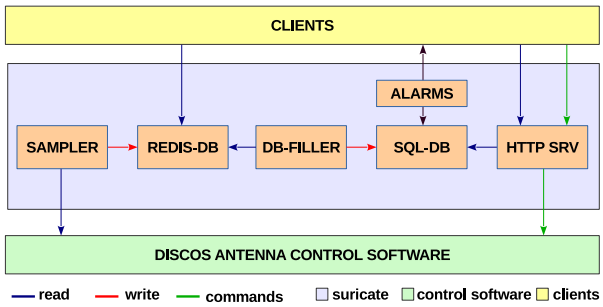

Figure 2: Suricate is a middleware which offers API and abstraction from the operating system and the programming language.

CONCLUSIONS

DISCOS is continuously evolving as new instrumentation is installed on the Italian radio telescopes. One of the major upgrades we are performing is the integration of new high-frequency receivers and data acquisition backends into the control software.

Such upgrades are made easier by the development choices the team previously made. The automatic deployment and software provisioning, together with the hardware simulators, allow us to add new features while keeping software robustness, speeding up maintenance and bug fixing. The Suricate middleware, thanks to its abstraction from the underlying control software, allows us to produce new applications (e.g. GUIs) exploiting also technologies that are not supported by the ACS framework.

ACKNOWLEDGEMENTS

The Sardinia Radio Telescope is funded by the Ministry of University and Research (MIUR), Italian Space Agency (ASI), and the Autonomous Region of Sardinia (RAS) and is operated as National Facility by the National Institute for Astrophysics (INAF). The Medicina and Noto radio telescope are funded by the Ministry of University and Research (MIUR) and are operated as National Facility by the National Institute for Astrophysics (INAF).

The Enhancement of the Sardinia Radio Telescope for the study of the Universe at high radio frequencies is financially supported by the Programma Operativo Nazionale (PON) del MIUR "Ricerca e Innovazione 2014-2020" Avviso D.D. n° 424 del 28/02/2018/ per la concessione di finanziamenti finalizzati al potenziamento di infrastrutture di ricerca, in attuazione dell'Azione II.1 - Proposta Progettuale PIR01_00010.

REFERENCES

[1] https://www.radiotelescopes.inaf.it/.

[2] G. Chiozzi et al., "The ALMA common software: a developer-friendly CORBA-based framework", in *Proc. SPIE*, vol. 5496, p. 205, September 2004.

[3] A. Orlati, M. Bartolini, M. Buttu, A. Fara, C. Migoni, S. Poppi, and *et al.*, "Design Strategies in the Development of the Italian Single-dish Control System", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, paper MOPGF110, pp. 330–333, http://jacow.org/icalepcs2015/papers/mopgf110.pdf, doi:10.18429/JACoW-ICALEPCS2015-MOPGF110, 2015.

[4] A. Orlati, M. Bartolini, M. Buttu, A. Fara, C. Migoni, S. Poppi, and *et al.*, "Evolution in the Development of the Italian Single-dish COntrol System (DISCOS)", in *Proc. ICALEPCS2017*, https://doi.org/10.18429/JACoW-ICALEPCS2017-THPHA014, 2018.

[5] DISCOS repository, https://github.com/discos

[6] M. Buttu *et al.*, "Rules of thumb to increase the software quality through testing", in *Software and Cyberinfrastructure for Astronomy III*, vol. 9913, p. 99130B, 2016. doi:10.1117/12.2230626

[7] Bartolini *et al.*, "DISCOS Project Status and Evolution Towards Continuous Integration", in *Astronomical Data Analysis Software and Systems XXVI ASP Conference Series*, Vol. 521, 2019.

[8] Righini S., Bartolini, M., 2016, Basie User Manual, IRA Technical Reports 492, http://www.ira.inaf.it/Library/rapp-int/492-16.pdf

[9] Carboni G., and Buttu M., "Discos simulators documentation", *INAF Technical Reports 82*, 2021. doi:10.20371/INAF/TechRep/82

[10] Ansible web site, https://www.ansible.com/.

[11] https://github.com/discos/deployment

[12] https://github.com/discos/suricate