Mem. S.A.It. Vol. 75, 282

Memorie della

Control software for the SRT. II

C. Migoni

INAF - Istituto di Radioastronomia, Sez. di Cagliari, Loc. Poggio dei Pini, Strada 54, I-09012 Capoterra (CA)

Abstract. The SRT project is not only the realization of a big radiotelescope but above all a modern instrument, with numerous technological devices which improve its characteristics and make it versatile for many needs. Its management is then more complex than that of other systems like the VLBI antennas in Medicina and Noto. The Advanced Common Software (ACS), developed by ESO for the ALMA project, has been identified as the suitable environment in which to connect the control programs that we are developing because it presents an excellent environment in which to exchange information and data.

1. Introduction

The guidelines of the Software working group[1] (G. Maccaferri, C. Migoni, M. Murgia, A. Orlati, F. Palagi, F. Schillirò) for the realization of the SRT control system were (and still are) the analysis of the existing systems (Medicina and Noto), the analysis of the new SRT system, the choice of new methodologies of designing system software, the choice of the operating system and languages of programming/developing software.

2. Existing systems (Medicina and Noto)

On the Medicina and Noto VLBI antennas every device (receivers, subreflector, active surface for Noto [Fig. 1]) is managed by specific "independent" programs which have, as a central archive for the exchange of information and data the Field System, the heart of the system. The Field System (Fig. 2) is an environment made some years ago by NASA/GSFC, which drives the VLBI terminal (like the new MARKV data acquisition terminal), the antenna and other equipment used during VLBI observations.

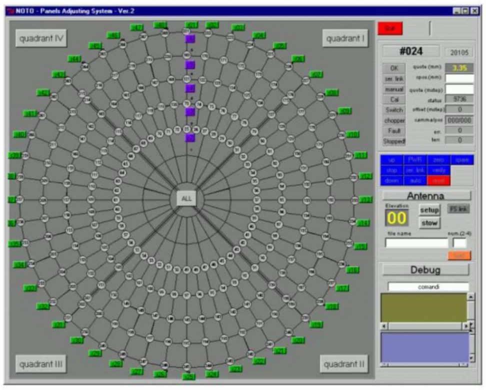

Fig. 1. Active Surface.

3. The SRT system

The SRT will be more complex than the Medicina and Noto antennas because it will have more devices which need to communicate and exchange data. Such devices are: Antenna

[1] `http://srtdoc.ca.astro.it`

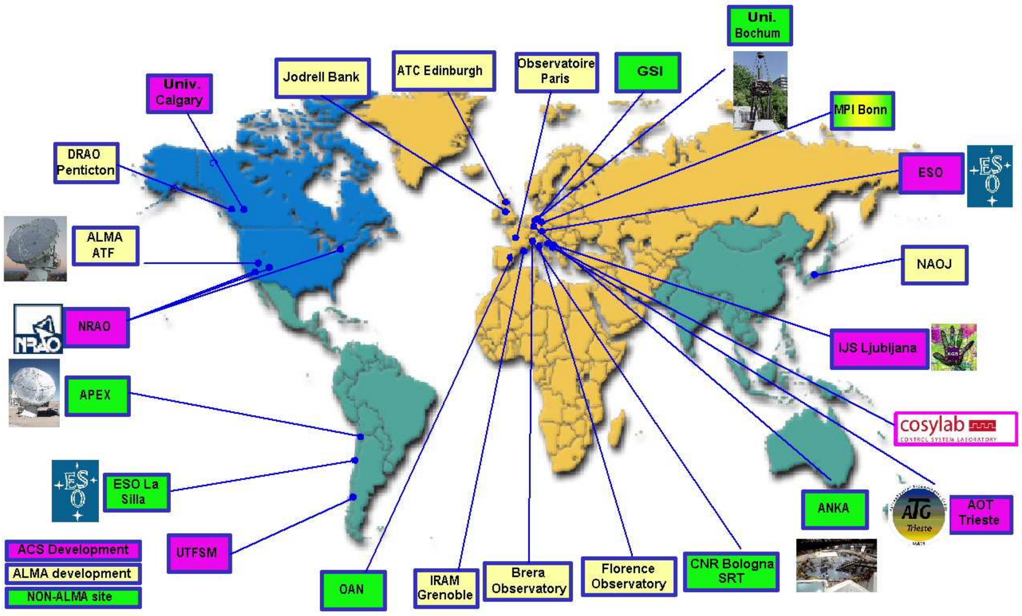

Fig. 3. ACS in the world.

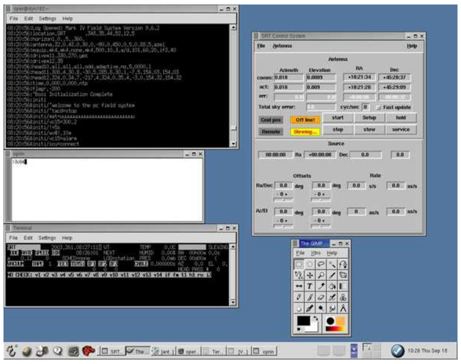

Fig. 2. Field System.

Control Unit (ACU), subreflector, active surface, receivers, beam wave guide (BWG), pointing and metrology systems; moreover we plan to integrate the Field System, create a data acquisition system and a realtime system for monitoring and data reduction.

4. Operating system and programming/developing languages

When we had to choose the operating system we put four constraints on it: stable, secure, well-documented and, obviously, economical. The natural consequence was the choice of Linux as platform for the new SRT control system. Regarding programming languages we decided on object-oriented languages because with them is it possible to divide a complex system like SRT into many components which we can describe, manage, develop, and modify without needing to review the whole system; for this, we chose C++ for managing and control algorithms and QT (a set of C++ API) for the graphical user interface for monitoring and interaction with the system.

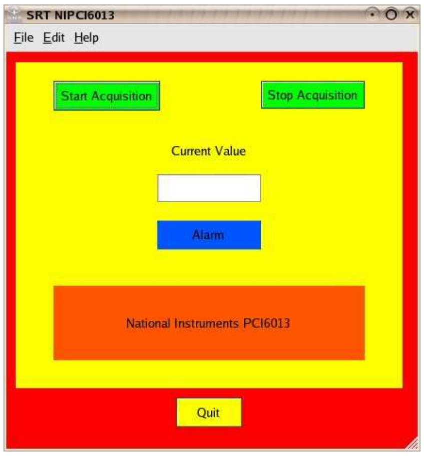

Fig. 4. National Instrument Data Acquisition Board.

Fig. 5. Medicina Antenna Control Unit.

5. ACS

ACS[2], developed by ESO, NRAO and Cosylab (Control System Laboratory), is a "general purpose" control system in the sense that it is used not only for the ALMA project and other radiotelescopes but also, for example, for the synchrotron ANKA, a particle accelerator in Germany (Fig. 3).

The principal characteristics of ACS are: object model of the devices (Distributed Objects, DOs), distribution of these DOs all over the system, data exchange between applications, database for the centralization and distribution of the system parameters, management of log messages, management of errors and alarms and remote control. ACS is essentially a mediator between high-level applications and specific management device software; this intermediation, useful to hide the system complexity, is made by the middleware CORBA (Common Object Request Broker Architecture) which is not a software but a set of specifics made by OMG (Object Management Group) which defines a versatile and multi-platform communication protocol between applications. Substantially, in a system that uses CORBA specifications, an application (written in C++, Java, or Python, for example) can interoperate with another application in every machine in the system, with every operating system and network (portability, scalability and interoperability). These DOs, the heart of ACS, have an architecture based on a Component/Container model; the Component contains the codes to represent and manage a hardware device (motor, temperature sensor, ACU, ecc.) and has management functions (methods), properties (azimuth, elevation, current, etc.), static characteristics (type, scale factor, limits, etc.). These properties and characteristics are inserted in a Configuration Database, a centralized database accessible by all the Components and Clients (applications) in the system. The Container is a process used by the Component to interface with the system; this interface is written using an Interface Definition Language (IDL), a file which contains methods and properties. A Client, through this IDL interface, can access functions and properties of the Component. Figs. 4 and 5 show our first prototypes of such applications.

6. IRAGRAPH library

Developed by Matteo Murgia (IRA, Cagliari), IRAGRAPH is a graphical library in which all principal instruments of scientific visualization like axes, curves, histograms are implemented. We have already inserted it in ACS and we want to use it to monitor physical properties, make maps, Fourier analysis, etc. This is illustrated in Figs. 6 and 7.

[2] `http://www.eso.org/~almamgr`

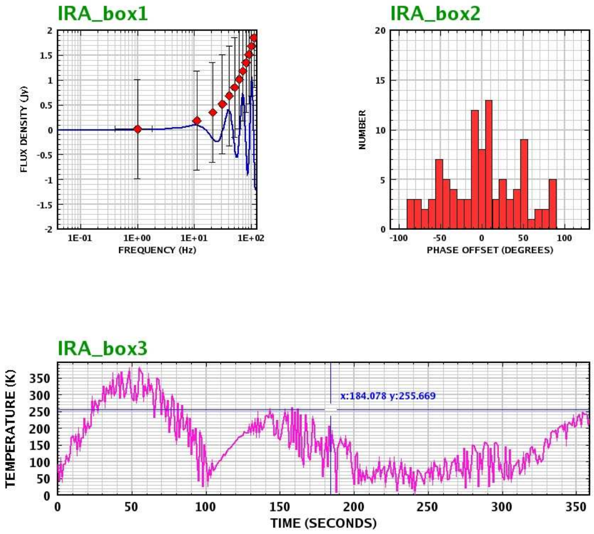

Fig. 6. Physical properties monitoring.

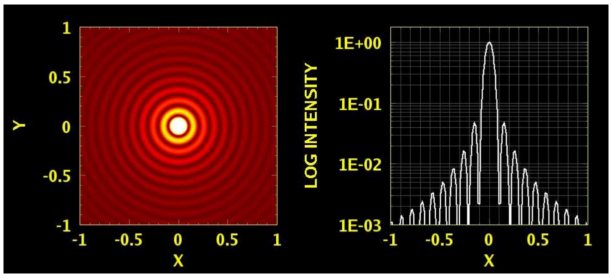

Fig. 7. Map example.