Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Software tools and preliminary design of a control system for the 40m OAN radiotelescope

Pablo de Vicente, Rubén Bolaño

Observatorio Astronómico Nacional, Centro Astronómico de Yebes, Aptdo. 148, 19080 Guadalajara, Spain

Abstract. The Observatorio Astronómico Nacional (OAN) is building a 40m radiotelescope in its facilities in Yebes (Spain) which will be delivered by April 2004. The servosystem will be controlled by an ACU (Antenna Control Unit), a real time computer running VxWorks which will be commanded from a remote computer (RCC) or from a local computer (LCC) which will act as console.

We present the tools we have chosen to develop and use the control system for the RCC and the criteria followed for the choices we made. We also present a preliminary design of the control system on which we are currently working. The RCC will run a server which communicates with the ACU using sockets and with the clients, receivers and backends using OmniOrb, a free implementation of CORBA. Clients running Python will allow the users to control the antenna from any host connected to a LAN or a secure Internet connection.

1. Introduction

The Observatorio Astronómico Nacional is building a 40m radiotelescope in its facilities in Yebes (Spain) which will operate between 2 and 110 GHz and will be finished in the spring of 2004. This telescope, designed by MAN Technologie and being built by Schwartz-Hautmont will be devoted to single dish and VLBI observations.

The antenna servosystem will be controlled by an antenna control unit (ACU) supplied by MAN. The ACU will run a real time OS (VxWorks) and will be commanded by a remote computer (RCC) and/or by a local computer (LCC) which will act as console. The OAN personnel is working on the definition of the ACU and the LCC tasks and on designing and implementing a control system for the RCC which will allow the operation of the antenna, receivers and backends. The ACU and LCC tasks are being implemented by MAN.

2. Selection of programming tools for our control system

As a first step we have chosen a set of programming tools for the RCC and for the clients. The selection criteria we have used are listed below:

- The tools should allow the feasibility of the project.

- The software should be free. The license should be GPL, LGPL, freeBSD or any other free license which allows to modify the code and share it with other institutions or persons without restrictions.
- The software should be widely used so that there is a community that supports it and should have a foreseen long lifetime which guarantees the project does not become outdated.
- The software should provide comprehensible documentation which allows to use it easily.

Based on our previous knowledge, on public documentation and after performing some tests to evaluate the fitness of the selected tools we have made a selection of tools.

The OS and distribution will be Linux/Debian. This is the distribution currently used in all hosts running Linux in the OAN and, although difficult to install, has a powerful package system which solves dependencies and allows simple upgrading. It also offers frequent security patches.

As programming languages we will use C++, C and Python. We wanted object oriented languages because they map to real objects better than other types of languages. Unfortunately in the OAN, like in many scientific environments, there is little expertise. Despite this fact we believe this is a good decision which will provide a longterm lifetime. Projects like ALMA, AIPS or GILDAS are using or will also use object oriented languages. C++ and C will be used in the server side, where speed is a requirement, and when controlling local hardware. Python will be used for the client side and where speed is not important.

We intend to use Python together with Qt for the GUIs. Qt is an excellent graphics toolkit written in C++, widely used and very well supported. PyQt is a Python package, which uses Qt as graphical interface. GUIs can be designed using QT-Designer, a visual editing tool which stores graphics forms in XML like format and allows to modify them easily. Its integration in the code is done using inheritance.

We have chosen MySQL because it is a fast, relational database which uses SQL and we have previous experience using it at the OAN. It will be used to store observations, engineer logs and queueing the observations sent to the telescope.

The control system requires multiple processes simultaneously working and being synchronized. The communication among them should be simple and seamless. We have chosen a CORBA implementation as the middleware to achieve this. In some cases we will also use shared memory and semaphores. CORBA allows the communication to be done independently of the location where the processes run. CORBA is also platform and language independent. We have chosen OmniORB 4.0.1 because it provides an implementation for C++ and Python, with basic services and its response time is rather low, being comparable to the best available CORBA implementations.

We have chosen CVS for software engineering. CVS is a system widely used which allows several developers to work in the same project using a central repository. It supports versioning, branching and tagging. There are graphical tools, including a web interface, which ease its use.

The documentation will be produced in PDF from OpenOffice and/or text files. The code will also contain comments which will be extracted using Doxygen.

3. Performed tests

We have tested some of the previous tools to verify its feasibility. We have created a set of C++ classes which wrap C libraries already developed at the OAN for controlling and reading RS232, GPIB ports, sockets and reading and writing on shared memory. A set of python modules for controlling serial ports have also been written and successfully used with our equipment. The Qt-Designer has been used to develop UI forms and a prototype was written to prove that the application responded to events both from the GUI environment and the network. In order to use Qt together with Python we have backported PyQt Debian packages from the unstable version to the stable one. We also tested CORBA by writing two servers and two clients, in C++ and in Python and demonstrated that both clients communicated successfully with any of the two servers. Finally we developed a graphical application which monitored a weather station using Python plus Qt and OmniORB to communicate from different remote clients to the server. We have also measured latency times in a busy LAN of the OAN to estimate the expected delays. A mean value of 1 ms was measured.

4. Architecture concept

The communication between the ACU and the LCC, a computer running Windows NT, will be done using sockets. The LCC will be controlled by the operators from 2 local control panels (LCP). All these hosts are to be delivered by MAN Technologie.

The ACU may be controlled either from the LCC or the RCC. The RCC will be a Linux/Debian host which communicates with the ACU using sockets. The receivers and backends will communicate with the RCC and with a server to store engineering logs using CORBA. The backend will also communicate with the data archiver and pipeline processor using CORBA. Finally the client host will control and monitor the whole radiotelescope through the RCC and the engineering log and data archiver hosts using CORBA. Fig 1A shows a schematic of the foreseen hosts and their communication channels.

Below we describe the characteristics of the running processes during a normal observation. Fig. 1B shows a simple diagram of the communication among them.

The control client will be written in Python and Qt. It will have a command line shell and a GUI. Variables in the shell and in the GUI will be common and therefore will be always synchronized. The shell will allow atomic and compound commands, interactively entered or using script files. The GUI will only allow compound commands, that is preconstructed observations. It will allow to add, modify and remove observations from the observations queue in the server and it will support plug-in applications to aid in the observations, coded as Python modules.

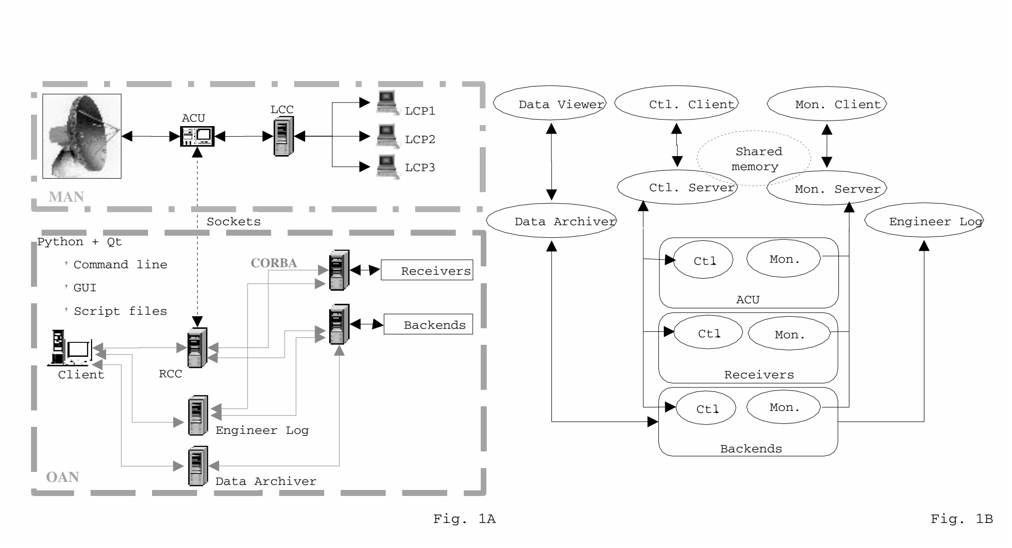

Figure 1. Architecture diagram.

The control server will be written in C++. It will manage the users database, authenticate users and establish user control according to predefined user priorities. It will also manage the observations database, queueing future observations and archiving completed ones. This process will transform the supplied coordinates, usually from a catalog, to apparent coordinates. Compound commands, representing observations, will be splitted into atomic commands for the ACU, receivers and backends. Atomic commands sent directly from the client will not be queued and will be immediately executed, unless a composed command is currently being executed.

The monitor server will be written in C++. It will listen to a network port using sockets where the ACU sends its status and store it in shared memory. It will receive status events using CORBA from the receivers and backends and store the parameters in shared memory. The control server will use the shared memory and semaphores to synchronize the observations with the status of backends and receivers.

The monitor client will be written in Python and Qt. It will connect to the monitor server using CORBA, request the content of shared memory and display information on the status of the antenna, receivers and backends.

The data archiver will be written in C++. It will receive the observed data from the backends and store them in a MySQL database using the MySQL C API.

The data viewer will be written in Python. This process will request data to the monitor server and will display live data using a graphical application. The GUI will be designed using Qt.

The engineer log will be written in C++ and will use MySQL. This process will receive status data from receivers, backends and auxiliary devices (weather station, maser, GPS), and store them in a database. The data will be viewed using a web interface or a customized graphic applications.