INTERNAL REPORT

Design of a tool for managing pulsar observations with the Sardinia Radio Telescope

Alessandro Corongiu

Report N. 35, released: 01/03/2014

Reviewer: N. D'Amico

Contents

Chapter 1

Introduction

SRT operations are generally performed through its omnicomprehensive suite `NURAGHE`, based on the `Alma Control Software` (ACS) framework. `NURAGHE` controls all telescope devices and coordinates them in all observations. Hence operators have the whole control of the telescope through a single global tool.

`NURAGHE` limits to one the number of the involved backends and, in the unfortunate case of failures, the observation is stopped. This policy is well understandable and meaningful in observations with a radio telescope. Nevertheless, pulsar observations may require an opposite policy, namely to use of several backends in parallel, either in the same or different data acquisition mode, and to not stop the data acquisition in case of failures.

These specific requirements for pulsar projects lead to the NON integration of all pulsar backends in the `NURAGHE` framework and to the design of a new management tool, specificly dedicated to pulsar observations. This tool is named **Srt ExpAnded Data Acquisition System** (`SEADAS`), and it has been designed in order to accomplish all observational requirements for pulsar observations.

Chapter 2

Tool design

The design of `SEADAS` started by thinking about a tool for controlling an instrument, generally speaking of any kind, that is required for an observation. It proceeded by imposing that all involved instruments have to be coordinated by `SEADAS`, then by giving to it the capability for coordinating the antenna setup and pointing with the setup and data acquisition progress of more than one backend in parallel. It finally ended by having in mind a tool for easily planning and managing an observing session.

2.1 Device control

In order to design the control of an instrument, it is necessary to consider it as composed of two main parts, namely:

1) The working device, responsible for doing the instrument's job;

2) A server, devoted to the direct control of the working device.

Figure 2.1 illustrates this simple scheme.

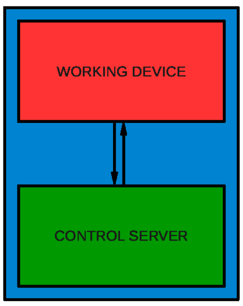

Figure 2.1: Instrument general scheme

The role of the control server is to run the `instrument's control software` responsible for setting and real time controlling the working device, and to write on its storage disks the acquired data. The instrument control software also accepts directives from external clients, i.e. softwares running on other servers, if a connection between the two servers is established through the mentioned codes. In this case the control software periodically reports to the external client the data acquisition status, progress and possible failures. Figure 2.2 illustrates the connection between `SEADAS` and the instrument.

Figure 2.2: Seadas to instrument connection

In this picture the role of `SEADAS` is to be the external client and, for each observation, gives to the control software commands for:

1) setting the instrument's specific data acquisition parameters;

2) starting/stopping the data acquisition

3) periodically quering the data acquisition status and progress, and the instrument's status.

In `SEADAS` have been implemented routines for sending the aforementioned commands to each instrument's control software and for reading any incomming message. The connection between `SEADAS` and the instrument plays a key role. For this reason these routines also monitor the connection status and react in case of problems.

2.2 Instruments coordination

`SEADAS` has been developed to be the instrument coordination tool. This simple requirement lead to the hierarchy scheme illustrated in figure 2.3.

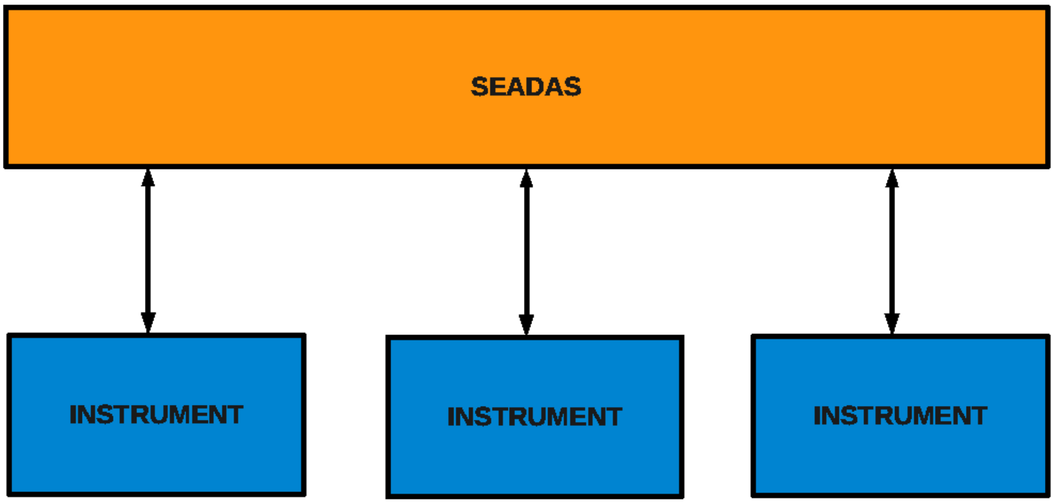

Figure 2.3: Seadas instruments hierarchy

The implementation of the adopted hierarchy on the network connections results in a total independence of each instrument from the others, since they are not directly connected. Their control software directly reports all events to `SEADAS`, which in turn analyses them and takes the opportune decisions.
The implementation of the adopted hierarchy on the code organization lead to a modular code. Each instrument is directly controlled by a `SEADAS` dedicated modulus, and all instruments moduli are in turn coordinated by a management modulus.

2.3 Antenna and backends coordination

Once established the instruments hierarchy, they have to be coordinated accordingly to their specific role in the observation. An observtion's pipeline can be simply summarized in the following points:

- Configure the antenna;
- Point the antenna to the source;
- Wait until the antenna tracks the source;
- Configure all backends;
- Start backends data acquisition;

- Periodically check:
    - the data acquisition progress;
    - the backends' system status;
    - the tracking status;
    - the antenna's system status;
- Stop the data acquisition;

This pipeline has been implemented in `SEADAS` accordingly to the specific organization of each control software with respect their way of accomplishing a specific requirement. Since for both the antenna and the `Digital FilterBank` backend (DFB) the configuration requires multiple commands in a precise order, the adopted solution has been to implement specific routines for building all necessary commands, and passing them to a secondary thread responsible for sending these commands to the target instrument. These sets of commands contain, after the configuration ones, the command for triggering the main action, namely for pointing the antenna or to start the data acquisition respectively for the antenna and the DFB. The case of the `ROACH based` backend is totally different. At the time of designing `SEADAS` it still was in its building phase and no control software was available yet. In collaboration with the backend engenieers team it has been decided to create a control software specifically designed to be managed by `SEADAS`. This software does not need multiple parameters for the backend configuration plus one for starting the data acquisition, but one single command that is at the same time the configuration and trigger command. Moreover, no query command are necessary from `SEADAS`, since the `ROACH` control software is set up to send to a connected client the data acquisition runtime parameters.
The aforementioned solutions and the fact that a parallel data acquisition by multiple backends does not need to be strictly syncronized, lead to rearrange the observation pipeline as follows:

- Configure the antenna and point it to the source;
- Wait until the antenna tracks the source;
- Separately configure each backend and start their data acquisition as soon as they are ready;
- Periodically check:
    - the data acquisition progress;

- the backends' system status;
- the tracking status;
- the antenna's system status;

- When the observation time is exipred stop the data acquisition;

2.4 Usability

A great attention has been payed in order to build an easy-to-use tool. Several experiences in pulsar observations with the Parkes Radio Telescope suggested to implement in `SEADAS` solutions already adopted in a similar tool, named `PMDAQ` (figure 2.4), created for managing pulsar observing session that required the use of the `Analogue Filterbank Backend` (AFB).

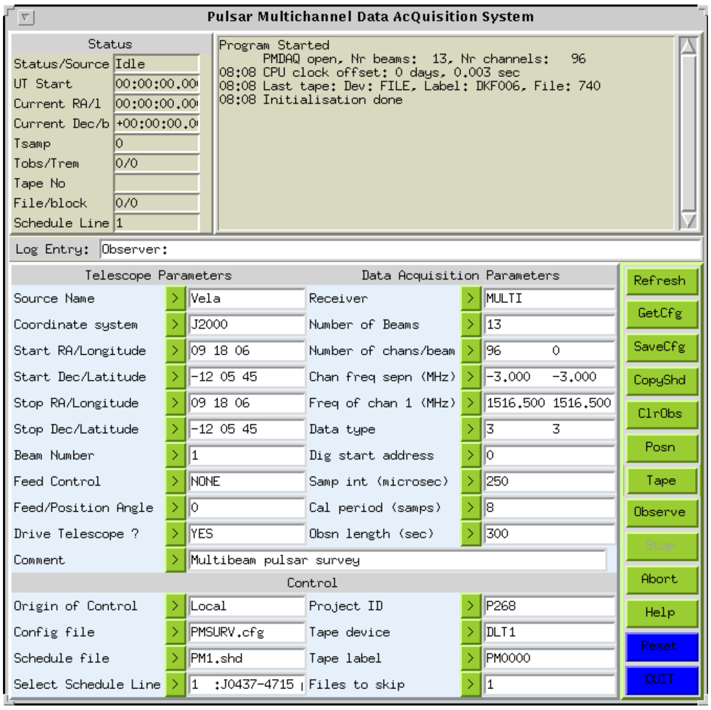

Figure 2.4: PMDAQ main window

In some cases these solutions have been simply imported as they were in `PMDAQ`, in some other they have been improved and in others reinterpreted accordingly to `SEADAS` peculiarities.

`PMDAQ` implemented solutions are:

1) the use of setup files: splitted and reorganized by following the already discussed hierarchy , they allow a better planning of the session and greatly simplify the structure of schedules' entries;

2) the schedules' structure: thanks to the use of setup files, the entry for each observation can be reduced to explicitly giving only the source name and configuration file only;

3) the sintax in both setup and schedule files: its basic structure, `[keyword] = [value(s)]`, makes both files easy to read and the keyword choiche allows an easy identification of its role;

4) a simply organized GUI: the main window is organized in frames, each of them devoted to a specific function. In `SEADAS` the GUI organization again reflects the hierarchy between `SEADAS` itself on one side and the instruments on the other.

5) an integrated source catalogue: `SEADAS` has an integrated pulsar catalogue for easlily finding the source of interests and loading its coordinates. Some pulsar parameters are displayed to help the user to find the most suitable source, in those cases when it's well clear the goal of the observation but not yet the object to observe;

6) the use of the `Schedule file` and the `Observing list` windows: loading a schedule does not necessary implies that observations have to be done in the schedule order. Users have the freedom to do so, but also to select some observations only by transfering them to the `Observing list` in any order. They have even the possibility of merging at runtime entries in different schedules whitout any file editing.

7) the flexibility of the `Observing list`: its content can be modified while the session is in progress. Exaclty like `PMDAQ`, `SEADAS` reads the first line only in the `Observing list`, when it has to start an observation in schedule mode, while all others can still be replaced, duplicated or removed depending on the session's evolution and user's runtime evaluations.

The implementation of the above mentioned `PMDAQ` solutions completely describe `SEADAS` usability and, as a matter of fact, its usage filosofy for those who have already used this Parkes's observting tool.

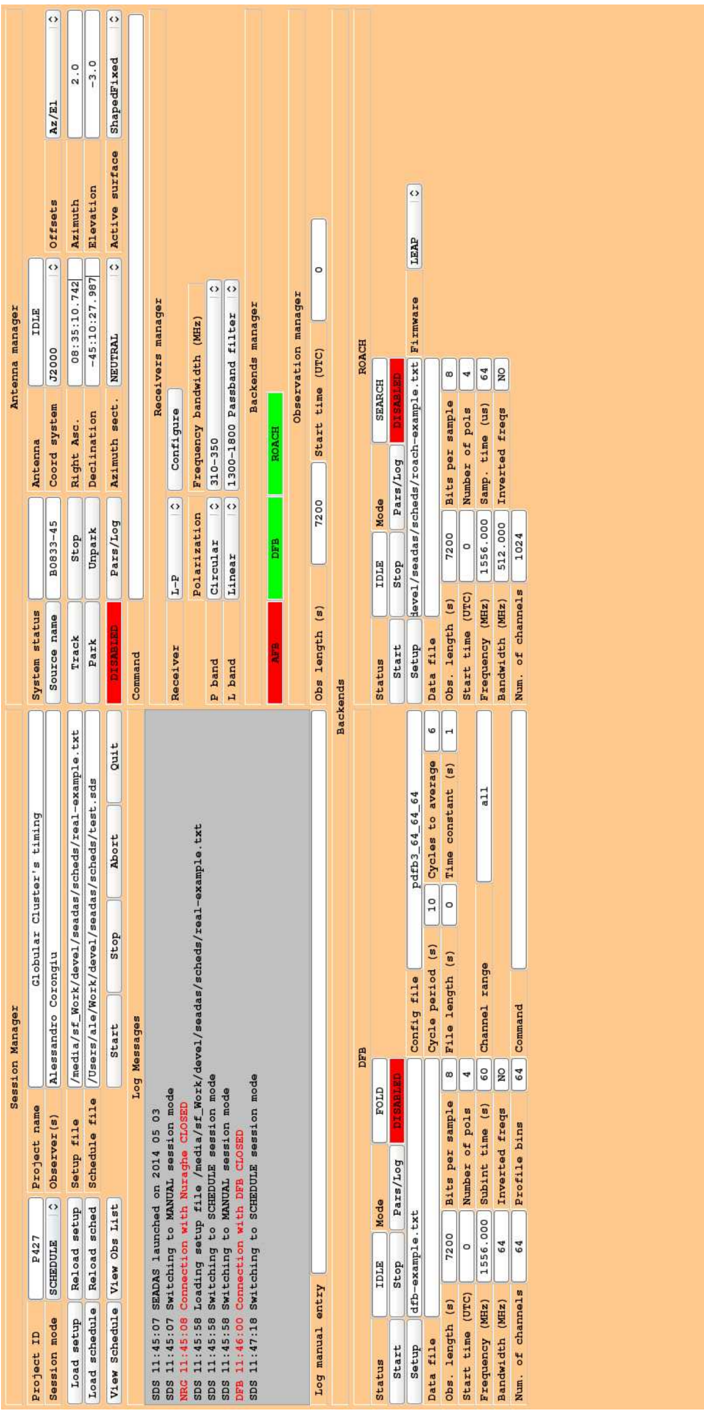

Figure 2.5: SEADAS main window

Chapter 3

Summary

This document reports the design of a software for managing pulsar observations with the Sardinia Radio Telescope, named **Srt ExpAnded Data Acquisition System** (`SEADAS`). The key points that drove its design have been the control structure for all involved instruments, their parallel control and coordination, the possibility of performing parallel data acquisition by multiple backends and its being an easy to use tool. All these requirement have been met by finding specific solutions, for the first three points, and by reimplementing already existing solutions widely tested and credited as valid for the last one. The result is a software that is modular in both its internal and graphical organization, that is friendly to be used and that allows the user to easy manage an observing session.