a) First a strong community of users. While the education and training of a new generation of astronomers is mainly the task of the member states, it is clear that ESO can help by furnishing opportunity for training and research to students and young postdoctoral fellows.

b) Second a commitment to support efficient operations. We have seen with NTT how important it is to properly commission and maintain sophisticated telescopes and instruments in order to obtain high scientific efficiency. We are working now to ensure that we will have a complete science verification, commissioning and maintenance programme prior to the VLT entering into operations.

c) The construction of new instruments. There are 16 possible focal plane instrument stations at the four 8 m telescopes. At the beginning 4 instruments will be completed and commissioned and 2 more will be in an advanced state of completion. If we were to provide 2 new instruments per year, the oldest instrument would be 15 years old at the building of the 16th instrument. Since ESO itself could certainly not provide more than a small fraction of this work, it is clear that much will have to be done by astronomical institutions in the member states to ensure that VLT will use front-line instruments.

d) The reduction and analysis of the data by the original observer must be made as effective as possible. ESO will support default options for calibration and reduction of the data which will guarantee a minimum set precision. ESO will also take responsibility for the archiving and distribution of the data to the community for additional research.

While the above activities will require a very considerable effort, it is also clear that attention needs be given to other sources of data on astronomical objects in a wide range of wavelengths. For this purpose we carry out a number of cooperative programmes with the European Space Agency (ESA) and the Space Telescope Science Institute (STScI).

The European Coordinating Facilities (ECF) is a joint ESA-ESO institution operated by ESO to manage the European participation in the use of the Hubble Space Telescope (HST). It has carried out its tasks with a high level of expertise which has resulted for instance in a joint successful development with STScI of the HST archival system. This expertise has been placed at the disposal of ESA to support other ESA missions such as ISO and will be fully utilized on VLT. ECF scientists have taken the lead in the scientific proposals for future ESA instruments for HST. It is possible that more extended ESA-ESO cooperative programmes could be undertaken in the future.

In a general way the collaboration between ground based and space observatories should be seen in the light of a modern view of astrophysical research in which the boundaries between specific wavelength domains or techniques are becoming irrelevant to the pursuit of astrophysical knowledge. For instance, there is a great deal of interest in the astrophysical community for the study of the large number of objects which have been or will be discovered from space missions, such as Rosat, ISO, XMM, etc.

In these collaborations ESO's strength resides in its ability to interpret and represent the scientific requirements of the community and to ensure the maximum scientific utilization of the data.

Upon successful completion of the VLT and VLTI, ESO will have developed the managerial, engineering and contractual capability needed to carry out large astronomical programmes in remote locations. Such capability will be unique in Europe. While the main task for the future will be the proper exploitation of VLT and VLTI, we expect that resources will also become available to undertake new major projects that may be advocated by the astronomical community. ESO will strive to continue to provide service to the community with excellence in astronomy as its goal.

TELESCOPES AND INSTRUMENTATION

The VLT Sequencer[1]

E. ALLAERT, ESO

Introduction

Software designed to control extremely complex equipment like the VLT is unavoidably itself also pretty complex. It contains impressive quantities of procedures, programs, libraries and other sorts of files. And even if from the very beginning a lot of attention is paid to the requirements of flexibility – with all these different instruments – it is difficult to foresee and manage all the possible combinations within monolithic executable programs. And if you want to add a feature to a program or change its behaviour, you have to edit its source code, compile it, link it with the necessary libraries and launch it again (many times just to see that the wrong line of code was modified). This is obviously a time-consuming exercise. Moreover, on top of a solid programming experience, also a profound knowledge of the VLT software structure is required to do that.

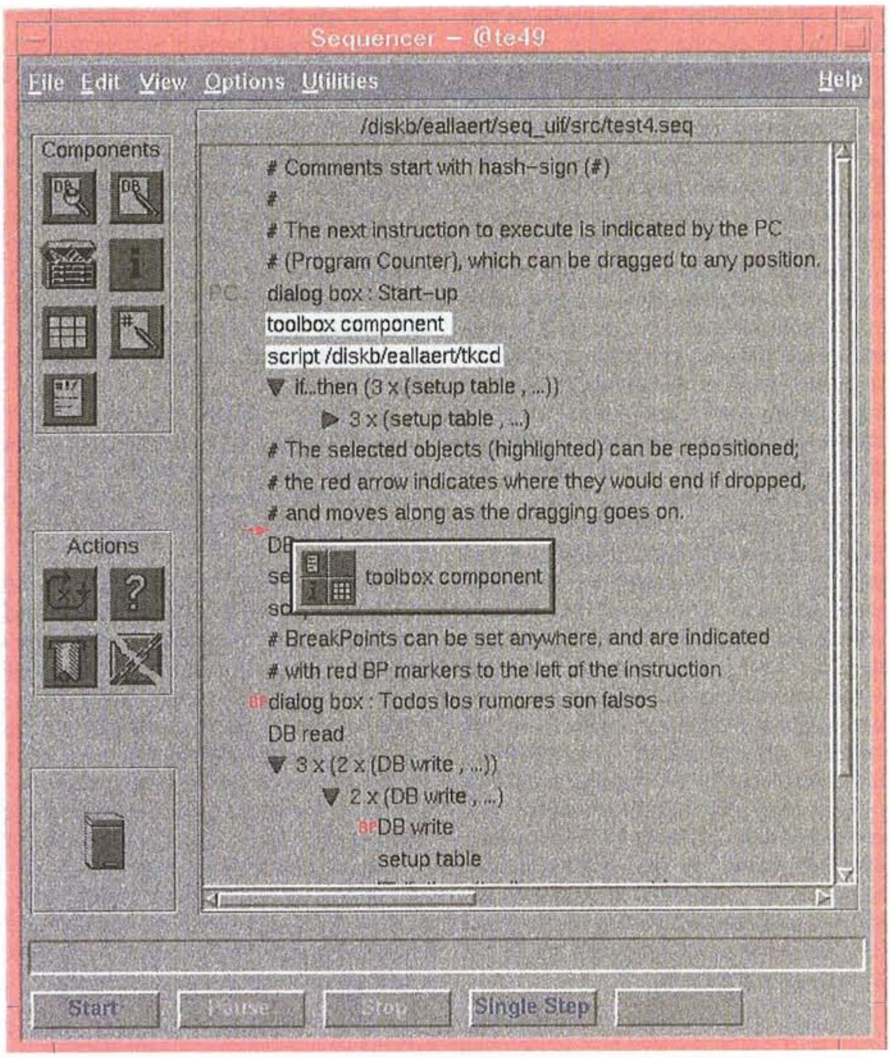
Specific user interfaces, which guide the users through the preparation and execution of their observations, already ease the pain offering some flexibility, as they allow the users to choose between various alternatives or to fill out specific values in forms. Still, this approach has its limitations, as changing such a graphical user interface (*GUI*) itself to e.g. add another parameter is again not trivial.

That means there is a real need for more and better tools to glue the basic commands and applications together, offering an easy means to control the VLT and its instruments at a higher level.

The Sequencer

Looking at how any telescope and instrument are operated, it is obvious that many commands are repeated in a certain sequence, sometimes with only slightly different values for the arguments. From an operational point of view, users differ one from the other as each of them repeats his own set of sequential commands. From this fact, the concept of Sequences was defined for the VLT control software; they are any

[1]This is part of a series of regular reports on VLT Control Software, started with issue number 81 of the Messenger (Raffi 1995)



```
/diskb/eallaert/seq_uif/src/test4.seq
   # Comments start with hash–sign (#)
   #
   # The next instruction to execute is indicated by the PC
   # (Program Counter), which can be dragged to any position.
PC dialog box : Start–up
   toolbox component
   script /diskb/eallaert/tkcd
   ▼ if...then (3 x (setup table , ...))
         ▶ 3 x (setup table , ...)
   # The selected objects (highlighted) can be repositioned;
   # the red arrow indicates where they would end if dropped,
   # and moves along as the dragging goes on.
   DB
   se
   sc
   # BreakPoints can be set anywhere, and are indicated
   # with red BP markers to the left of the instruction
BP dialog box : Todos los rumores son falsos
   DB read
   ▼ 3 x (2 x (DB write , ...))
         ▼ 2 x (DB write , ...)
             BP DB write
                setup table
```

Figure 1: *A snapshot of the Sequencer's prototype GUI.*

combination of known commands. They can contain instructions controlling the flow of execution, offering a great deal of flexibility and intelligence. Remark that this definition covers a very broad range of cases. E.g. a set of low-level commands dealing with part of the execution of a single exposure can be grouped together as a sequence, which then can be considered 'a high-level command. On the other hand also the combination of several exposures can be a sequence, as only a few high-level commands are needed to execute a single exposure.

The *Sequencer* is the engine around the concept of these sequences. Depending on the level of programming awareness and interest of its users, we can subdivide this product into two distinct components:

1. a scripting language, which allows developers and operators to glue their commands and procedures quickly and efficiently together. The purpose may be either to implement high-level functionality or to to create complex utilities, including GUIs. These scripts generally relieve users from repetitive tasks and manual intervention.
2. a tool to assist astronomers in the preparation and execution of sequences of commands and observations. A primordial aspect of this tool is its user-friendliness: it must be very intuitive to use, and should avoid to expose its users to the complexity of the underlying software components, scripting language, etc.

The Scripting Language

The Sequencer's scripting language is based on a free software package, namely Tcl/Tk (Ousterhout 1994, Welch 1995) Tcl stands for "Tool command language"; it is a simple programming language, while *Tk* provides an X11 ToolKit or library of widgets (graphical objects similar to those of other GUI toolkits like Xlib or Motif). This permits us to make windowing applications as simple scripts, hiding many of the details faced by C programmers. These scripts are interpreted by a shell built around the Tcl/Tk library.

Tcl was chosen as the scripting language for the VLT Sequencer as many of its features could be mapped into our requirements. Also, since its conception it has enjoyed a very good acceptance in the scientific and astronomical community, and has in the meantime an impressive user base.

One of Tcl's nice features is its extensibility, i.e. it is relatively easy to add more commands to the core. Of course this is something most specific applications based on Tcl have to do, and the VLT Sequencer shells are no exception to that. The resulting interpreter incorporates a number of extensions, including our own 35 or so commands to interface the VLT Central Control Software (a set of common services like database access and message system – see Raffi, 1995). It contains all necessary flow control commands to allow sequences to behave intelligently. However, it is certainly not our intention to map every procedure call available in the VLT C-libraries into a corresponding script-command. Bear in mind that the purpose of the Sequencer is to provide some *glue*; it is not meant as a substitute for C-programming. Sequencer scripts are interpreted, so they can never run as fast as equivalent C programs.

A number of VLT applications have already been built using this scripting language. The most complex and notorious application of this kind is at present the Panel Editor. This is a tool to graphically create panels which conform to the *ESO GUI Common Conventions*. Other applications are several engineering interfaces, e.g. to configure or diagnose motors.

The Sequencer Tool

The Sequencer Tool itself, presently in a prototype status (see Figure 1), is also programmed in the Sequencer scripting language. The goal is to permit simple visual programming of sequences.

These are composed of fundamental building blocks (*components*), on which control flow *actions* can be performed. The use of the Panel Editor provides a look-and-feel similar to other VLT applications, in areas like menus, short-help text, etc. Drag-and-drop functionality and other features aim at making this tool even more intuitive.

Looking back at how this tool is made and what it produces we see an interesting closed loop: the tool creates scripts in the same language with which it was made in the first place. The fact that part of the Sequencer Tool was made with the Panel Editor – also a script producing scripts – seems to make it even more self-contained.

What's in the labs

The Sequencer interpreter is based on the Tcl/Tk core as explained in 3. The versions we use are 7.3 for Tcl and 3.6 for Tk. On top of these the Sequencer interpreter includes currently the following extensions:

1. *Extended Tcl/Tk*: this extension adds several commands useful in a Unix context; its version has to match the one of Tcl/Tk, as it patches this core.
2. *BLT*: the Bell Labs Toolkit adds a few commands to the Tk library. Presently at version 1.7.
3. [*incr Tcl*] or *iTcl*: this extension provides an object oriented programming environment in Tcl/Tk, which allows and promotes organisation of the scripting code in a much cleaner way than plain Tcl does. iTcl is at version 1.5.

A few months ago new versions of Tcl and Tk were released (7.4 resp. 4.0), which provide more functionality, better Motif compliance and tons of other modifications and improvements. Of course the Sequencer interpreter will be updated to these new releases of Tcl/Tk, but we also need matching versions of all the extensions included in our interpreter. In particular we are waiting for the 2.0 version of iTcl[2], which should bring a major performance improvement as a bonus. That should manifest itself a.o. in all applications created with the Panel Editor.

Once we have the needed extensions, we will still have to address the compatibility issues: several VLT applications have been implemented as Sequencer scripts, and they may be affected by modifications in Tk's behaviour or syntax.

We expect to have this update for the Sequencer interpreter ready for internal use early 1996, and to release it externally with the next subsequent VLT Common Software Release.

References

Ousterhout J., 1994, *Tcl and the Tk toolkit*, ISBN 0-201-6337-X.
Raffi G., 1995, *The Messenger* **81**, 5.
Welch B., 1995, *Practical Programming in Tcl and Tk*, ISBN 0-13-182007-9.

[2][incr Tcl] version 2.0 should be available by the time you read this.

The VLT CCD Detectors Control Software[1]

A. LONGINOTTI, C. CUMANI, P. DUHOUX, ESO

Introduction

Charge Coupled Devices (CCD) are currently by far the most widely used type of detectors in Astronomy. It is foreseen to have in operation at the VLT about 40 technical CCD cameras for auto-guiding, field viewing and wavefront sensing, and probably more than 10 scientific CCD cameras for instruments working in the optical spectral range. Because of the critical role it plays, it has been decided that the CCD Software Package should be the very first VLT application to be built and tested on top of the VLT Common Software. Because of that, the CCD Software has provided, together with the NTT upgrade software, a considerable feedback to the VLT common software in terms of bug detection and suggestions for improvements.

It has also been decided to put in the development life-cycle a prototype stage, to allow a better evaluation of the validity of the whole camera concept, including, of course, its control software. A field test has been successfully performed January 1995; from that time efforts have been concentrated in adding functionality and bringing the whole package to VLT standards, both in terms of software quality (quality of code, documentation, configuration control) and of general rules applicable to all VLT software applications (usage of standard commands, libraries, etc.).

The first release of the CCD Control Software, originally planned for August 1995, has been recently sent out (October 1995). It contains a sub-set of the foreseen functionality, and does not support technical CCD cameras yet (this will be done with the next release). On the other hand, it defines the complete interface towards external software.

Baseline development criteria

The development of the CCD Control Software has been based on the following criteria:

- It must fit in the general VLT control architecture (distributed system consisting of Workstations, for high level operations and user interface, and Local Control Units, for real-time and sub-system hardware related operations). For more information, see the article of G. Raffi about the VLT control software in the Messenger, issue September 1995.
- It must use, wherever possible and compatible with the performance requirements, components provided by the VLT common software.
- It must provide a programmatic interface to all packages (instruments and telescope control software) using CCD cameras, but also be able to work as a stand-alone simple instrument, mainly for test purposes.
- It must interface with the already defined and developed transputer-based CCD controller box, also called Array Control Electronics (ACE).
- CCDs being a technological area in very rapid progress (bigger and faster chips are continuously coming up on the market), it must have an interface to the hardware as flexible as possible, in order to accommodate possible upgrades of the ACE-based or totally new CCD controllers, keeping the bulk of this package independent from the particular controller used.
- In order to simplify its maintenance, the CCD software must be one package, providing the functionality required for both technical and scientific cameras. Differences in the control software between these two categories must reflect only the differences in

[1]*This article is part of a series of regular reports on VLT Control Software, which started with the previous issue of the Messenger. (Raffi 1995)*