STUDY OF PORTABILITY OF VLT INSTRUMENTATION SOFTWARE TO ACS

Roberto Cirami, Paolo Santin, INAF-OAT, Trieste
Gianluca Chiozzi, Antonio Longinotti, ESO, Garching bei Muenchen

Abstract

The Very Large Telescope (VLT) will remain in operation most probably for at least two more decades. Being the software technology currently used at the VLT more than one decade old, the maintanability of such a complex system might become a critical issue. The ALMA Common Software (ACS) is based on newer technology. Following this consideration, one of the obvious options to improve the maintainability of the VLT Software would be to port it, or parts of it, to ACS. This would allow optimizing maintenance resources for both VLT and ALMA Software, eventually making available resources for new ESO projects, such as E-ELT. Because of operational constraints, this can only be achieved gradually, possibly starting with new VLT sub-systems. In the year 2004 a pilot project has been started to study the effort needed to replace standard components of the VLT Instrumentation Software with ACS based ones. Starting from a simple instrument created from the VLT Template Instrument and entirely based on the VLTSW, we have replaced the core of the Observation Software (OS) with an ACS based equivalent. The purpose of this paper is to summarize the work done and draw some conclusions.

INTRODUCTION

In the year 2004 an experimental project has been started, to study the efforts needed to replace standard components of the VLT Instrumentation Software with ACS based ones. The Observation Software (OS) package of a simple instrument created from the VLT Template Instrument [1] and entirely based on the VLTSW has been replaced with an ACS based equivalent. The Instrument Control Software (ICS), as well as all the other packages, internal to the instrument, such as the Detector Control Software (DCS), and external ones, interfacing to the instrument, such as the Telescope Control Software (TCS), the Broker for Observation Blocks (BOB) and the Data Flow Software (DFS), remain unchanged, i.e. fully VLTSW based.

REQUIREMENTS

The initial requirements have been formulated as prototype evaluation criteria:

- **Reusability**. It must be possible to interface with a minimum effort and (almost) no specific coding an existing VLT application to an ACS based one, using a generic VLT Wrapper.
- **Accessibility**. Any ACS application should be able to access VLT On-Line database (OLDB) functionality, receive a VLT error stack and interface with the VLT message system, using code produced for the prototype, without having to write specific code.
- **Compatibility**. The ACS based parts of the prototype must produce the same results as the initial pure VLT implementation.
- **Efficiency**. A comparison of the effort needed to produce some functionality (e.g. OS main functionality) within the VLT environment and ACS shall be done. A comparison of the two learning curves is desirable.
- **Performance**. An evaluation of performances between the VLT and the ACS version of the prototype instrument will be done (e.g. how long it takes to execute the self test OB).

DESIGN AND IMPLEMENTATION

An ACS-based OS prototype has been implemented following the Component/Container paradigm adopted by ACS [2] [3]. This OS, being embedded in a VLT environment (see Fig. 1), receives pure VLT commands (e.g. from BOB) and forwards them to different subsystems/applications either VLT or ACS-based ones. A VLT Wrapper allows forwarding the commands to VLTSW based subsystems. For the prototype, a new VLT software instrument has been created starting from the XXXX template instrument, which contains three VLT subsystems (ICS, FIERA1 and FIERA2). All the replies as well as any error coming from the VLTSW based subsystems are managed by the prototype.

For the prototype implementation the following software releases have been used:

- VLTSW JAN2006
- ACS 5.0.2

The main ACS prototype components are described in the next subsections.

Command Dispatcher

The Command Dispatcher is implemented as an ACS Characteristic Component. It receives pure VLT commands and activates dedicated ACS Characteristics Components (Command Components) for each received command (e.g. SETUP, START, WAIT, …). The correspondence between the VLT command and the related Command Component is specified as a CDB (ACS database) entry of the Command Dispatcher and stored as an internal STL map. The Command Dispatcher contains an ACS callback for managing the VLT replies.

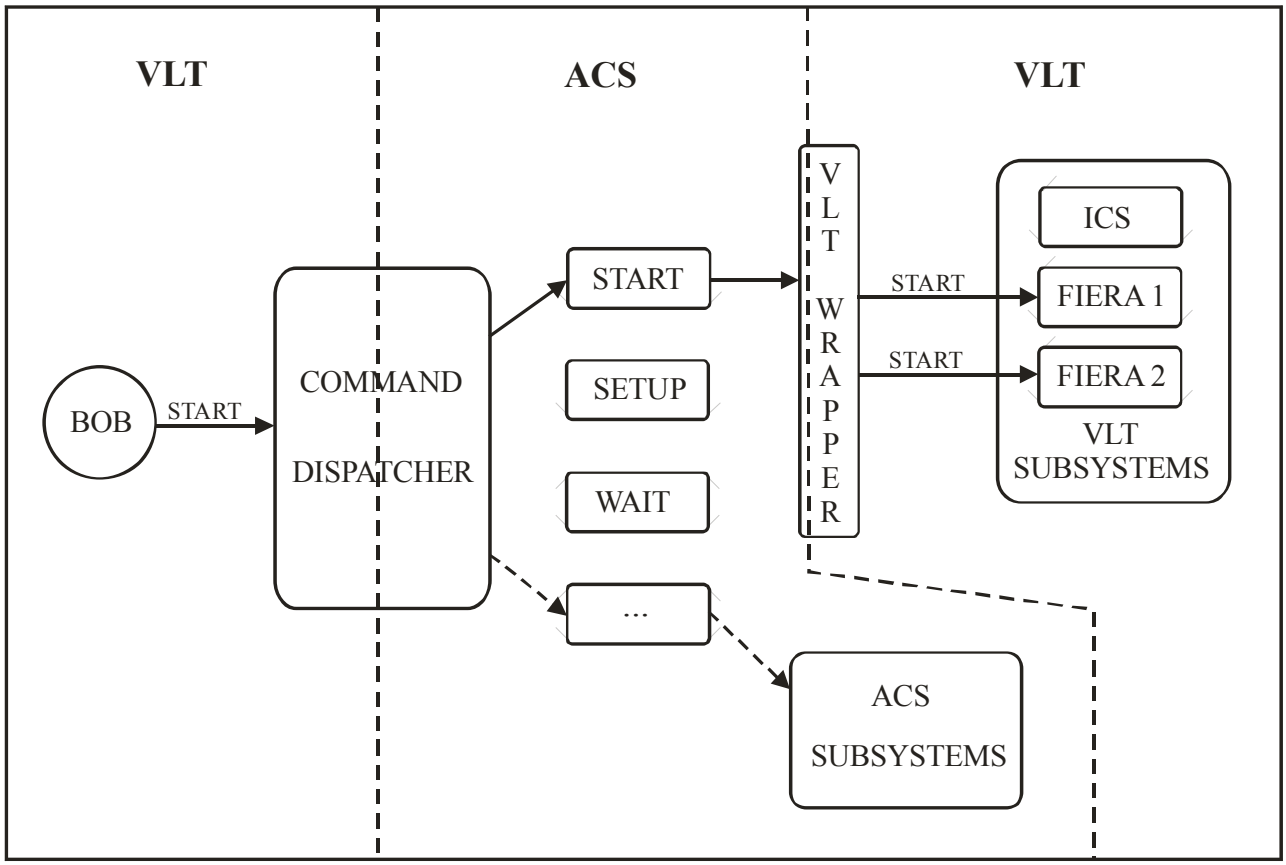

Figure 1: Overall prototype representation. The commands (e.g. START) are sent by BOB to the ACS-based OS. The VLT Wrapper allows forwarding the commands to VLTSW based subsystems.

ACS-Based OS (Command Components)

In the prototype, the pure VLT OS is replaced by an ACS-based OS. It is represented by the whole set of Command Components, corresponding to the commands specified in the Command Definition Table (CDT).

Each Command Component inherits from a common base class which implements common methods and provides the ACS callback needed for managing the ACS/VLT replies. Each Command Component implements a well defined common interface. This interface, called by the Command Dispatcher in response to an incoming VLT command, implements the logic specific to each command (e.g. for the START component it splits the incoming command into START commands to be forwarded to the specific VLTSW based instrument subsystems, etc.), and forwards the command to the ACS or VLT subsystems.

VLT Wrapper

The VLT Wrapper represents a layer between the ACS part of the prototype (Command Dispatcher and Command Components) and a pure VLTSW based subsystem. The VLT Wrapper exposes a well specified interface to the ACS part of the prototype. This interface is called by the Command Components and forwards the specific command to the different VLT subsystems (ICS, FIERA1, FIERA2). To communicate with the VLT subsystems the VLT Wrapper uses the VLT message/error systems. Internally, the Wrapper manages the VLT subsystems replies and errors, and communicates with the ACS callback of the Command Components.

The VLT Wrapper is totally application-independent. Any ACS application can use the VLT Wrapper to communicate with VLTSW based subsystems provided it uses its interface.

VLT Container

An ACS Container modified to allow for the VLT environment provides the infrastructure for managing the life cycle of all the aforementioned ACS Characteristic Components. In this way the VLT Container is seen by the VLT environment as a single application.

VLT Replies Handling

To handle the VLT replies, the Command Dispatcher and the Command Components activate pure ACS callbacks. The prototype callback mechanism works in the following way:

1. When a VLT subsystem (ICS, FIERA1, FIERA2) sends back a reply, it is caught by the VLT Wrapper.
2. The VLT Wrapper informs the appropriate Command Component (e.g. START if a START command has been issued) of the reply, calling the ACS callback of the Component.

3. The ACS callback of the Command Component calls the Component itself for managing the reply.
4. After managing the reply the Command Component calls the ACS callback of the Command Dispatcher.
5. The Command Dispatcher sends back the reply the VLT caller (e.g. BOB) is waiting for.

Error Handling

When an error occurs in any of the VLTSW subsystems its error stack is included in an ACS error and propagated back to the caller. Through the error handling mechanism the VLT error stack is fully encapsulated in an ACS error structure and propagated back to the each prototype component. At each step it is possible to check the error and take the appropriate action.

ACS Files Automatic Generation

For every command present in the Command Definition Table (CDT) of a VLT application, it is necessary to implement a specific Command Component. In the ACS environment, each Command Component requires the presence a certain number of files, from the implementation files to the xml files needed from the ACS database (CDB). For each Command Component these files have the same skeleton.

In order to automate the generation of these files a Tcl script has been implemented. The script reads the list of commands present in the CDT (from memory, if the CDB is installed in it, or from a CDT file) and creates automatically for each command the required files.

OLDB Interface

Any ACS application can access the VLT OLDB using only ACS code. To make it possible an ACS-VLT interface for the OLDB has been implemented.

This interface has been implemented as an ACS Characteristic Component. For each VLTSW type (long, double, …) this interface exposes two methods (e.g. readLong and writeLong).

The read method takes as parameter the OLDB address and returns the value. The write method accepts two parameters: the OLDB address and the value to be written.

This interface avoids the use of pure VLT code when an ACS Component tries to access the OLDB.

CONCLUSIONS

The general conclusions can be summarized as follows:

- Implementing a VLT Instrument using the core functionality of ACS is technically feasible.
- ACS and VLTSW are quite different. Embedding ACS based packages in the VLT environment, although technically feasible, requires intermediate Software layers, which add further complexity to the system, thus deteriorating its maintainability.
- At the present stage of the VLT and ALMA projects and related Software, the implementation of new VLT Instruments should still be done in the current standard way, i.e. not using ACS.
- The extension of this prototype, e.g. to ICS or DCS, can and actually should be taken into consideration if and only if ACS is going to be used for the next big ESO project (E-ELT) and the VLT, or parts of it, is going to be used as test bench for the new system (similarly to what has been done with the NTT at the beginning of the VLT project).

Concerning the initial requirements, the conclusions for each evaluation criteria are:

- **Reusability**. A generic, totally application-independent VLT Wrapper has been implemented. This Wrapper can be used with any ACS application, as long as it implements its interface protocol.
- **Accessibility**. Any ACS application can access the VLT OLDB functionality using the OLDB interface. The VLT error stack is automatically included in an ACS error by the VLT Wrapper and passed back to the ACS application. The VLT message system is used by the VLT Wrapper to communicate with the VLT subsystems, hiding its use to the ACS application.
- **Compatibility**. The ACS prototype behaves in the same way as the VLT original application (with reduced functionality).
- **Efficiency**. Thanks to the huge amount of available common Software and templates, the development time in the present VLT environment is much faster. A correct and fair comparison should however be done between the current ACS status and the VLTSW before the VLT first light. If we make this comparison, it is our opinion that in absolute terms the code production in the ACS context is much more flexible and effective.
- **Performance**. By running a simple test OB, no substantial differences in performance have been measured between the full VLTSW implementation and the final ACS based prototype. However, due to the simplicity of the prototype, this cannot be considered a complete comparison between the two implementations in terms of performance.

REFERENCES

[1] "VLT template Instrument User Manual", VLT-MAN-ESO-17240-1973, Issue 5, January 2005.

[2] G. Chiozzi et al., "The ALMA common software: a developer friendly CORBA-based framework", Proc. SPIE Vol. 5496-23, Astronomical Telescopes and Instrumentation, Glasgow, June 2004.

[3] H. Sommer, G. Chiozzi, "Container-component model and XML in ALMA ACS", Proc. SPIE Vol. 5496-24, Astronomical Telescopes and Instrumentation, Glasgow, Scotland, June 2004.