VST (VLT SURVEY TELESCOPE) CONTROL SOFTWARE DESIGN

D. MANCINI, AND M. BRESCIA
Osservatorio Astronomico di Capodimonte, via Moiariello, 16, 80131 Napoli, Italy

1. Introduction

The VST (Very Large Telescope Survey Telescope) is a 2.6 m wide field survey Alt-Az telescope to be integrated within the array of the VLT, (Very Large Telescope), already installed at Paranal (ESO-Chile). The VST Software is a part of the VST Programme, devoted to control and drive the telescope and the related instrumentation at the Cassegrain focus.

2. Telescope control system overview

The basic concepts which are at the base of the VST software development are related to the following constraints:

- The VST software is based on the existing well tested VLT TCS (Telescope Control Software).
- The VST software will consist of a number of packages which, individually, will be complex real-time systems.
- The packages will be subdivided into modules, developed and tested individually, and then integrated together, with the hardware components, and, except for the first release, with the existing system.

The VST control architecture consists of two workstations (WS), respectively used for TCS and Guide/Sensing, high level management and of some Local Control Units (LCU), dedicated to subsystem control. All these processing units are connected together via Local Area Networks (LANs).

The lowest level interface is obtained at level of LCUs. The task repartition is listed below:

- TCS LCU#1 AZ axis Astronomical computations
- TCS LCU#1 AZ axis corrections from guiding system

- TCS LCU#1 AZ axis reference trajectories planning
- TCS LCU#1 AZ axis encoders data reading and control
- TCS LCU#1 AZ axis position loop control
- TCS LCU#1 AZ axis tachometers data reading and control
- TCS LCU#1 AZ axis speed loop control
- TCS LCU#1 AZ axis preload torque and torque command computation
- TCS LCU#1 AZ axis motor power amplifiers control
- TCS LCU#1 AZ axis high speed diagnostic for drivers, motors, tachos, cooling and power
- TCS LCU#1 AZ axis emergencies, faults and warnings management
- TCS LCU#1 AZ axis interlocks management
- TCS LCU#1 AZ axis temperature data readout
- TCS LCU#1 AZ axis rack and motor cooling control
- TCS LCU#1 AZ axis power supply control

- TCS LCU#2 EL axis Astronomical computations
- TCS LCU#2 EL axis corrections from guiding system
- TCS LCU#2 EL axis reference trajectories planning
- TCS LCU#2 EL axis encoders data reading and control
- TCS LCU#2 EL axis position loop control
- TCS LCU#2 EL axis tachometers data reading and control
- TCS LCU#2 EL axis speed loop control
- TCS LCU#2 EL axis preload torque and torque command computation
- TCS LCU#2 EL axis motor power amplifiers control
- TCS LCU#2 EL axis high speed diagnostic for drivers, motors, tachos, cooling and power
- TCS LCU#2 EL axis emergencies, faults and warnings management
- TCS LCU#2 EL axis interlocks management
- TCS LCU#2 EL axis temperature data readout

- TCS LCU#2 EL axis rack and motor cooling control
- TCS LCU#2 EL axis power supply control

- TCS LCU#3 Cassegrain rotator axis control
- TCS LCU#3 M1 axial active pads control
- TCS LCU#3 M2 hexapod control system functions management
- TCS LCU#3 high speed diagnostics management
- TCS LCU#3 emergencies, faults and warnings management
- TCS LCU#3 temperature data collecting
- TCS LCU#3 rack and motor cooling control
- TCS LCU#3 power supply control
- TCS LCU#3 auxiliary subsystems (M1 cover, baffles, auxiliary systems) control
- TCS LCU#3 ADC control

- TCS LCU#4 Hydrostatic Bearing system control (ESO & VST team)

- LCU#4 Telescope Autoguiding system control
- LCU#4 Technical CCD Camera Head and Array Control Electronics (ACE) management
- LCU#5 Shack-Hartmann Image Analysis system control
- LCU#5 Technical CCD Camera Head and Array Control Electronics (ACE) management

The high level VST Telescope Control Software (TCS) performs the following tasks:

- Controls and monitors the telescope
- Interfaces to users and other VST packages
- Interfaces to and gets data from star catalogues

The TCS makes extensive use of VLT Common Software, which provides common services, utilities and tools. In particular, generic services like logging, error handling and alarm handling are already implemented by the VLT Common Software. It is a general enough software package that can be reused on the VST to a large extent. Differences will exist at a lower level in the interfaces with control electronic subsystems. Therefore there is a commitment by the VST team to re-use as much as possible, in agreement with ESO, existing VLT TCS control software. The VST TCS software can be grouped in four categories:

- User interface
- Coordinating software
- Subsystem application software
- Special interface software

The subsystem application software implements all the functionalities that can be performed locally in a LCU, without any knowledge about other system components. The coordinating software is hierarchically above the subsystem software, in the sense that it performs actions of combined, coordinating nature, and it often uses one or more subsystems to execute its actions. It is also the software running on WS. The special software interface is based on a group of libraries to access via TCS external systems, like star catalogues. A particular role in the VST TCS is performed by the coordinating software, that provides a public interface for external applications requiring services to TCS, through several functional software modules, composed basically by specialized processes, running concurrently on TCS WS. The main modules belonging to coordinating software are presetting, tracking, autoguiding, enclosure control and active optics. All inter-module and external communication makes use of the Central Control Software (CCS) message system. Each module has a Command Definition Table (CDT) which is the only one normally used for all inter-module communication and which is also used by the TCS user interface panels. In the general TCS module architecture, all the modules are implemented through one or more independent process. If a module is made up of more than one process, only one of them is the control process, i.e. is responsible for providing the public interface and to receive commands from other modules and to send back the corresponding replies. In order to respect these requirements, every process is designed in an event-driven way and the implementation is based around the event-handler provided by the evhHANDLER class, that is the core component of the ESO WS software library. All the above guidelines and features described will be completely adopted on the VST TCS to match all the standardization requirements expected by ESO.

3. TCS general architecture

The Telescope Control Software for the VST has been designed and implemented using Object Oriented methodologies. The workstation components are implemented in C++ and are based on the ESO EVH toolkit. The TCS package provided by ESO is composed by several modules referred to the telescope subsystems to be handled by the Telescope Control System. The general functional architecture is shown in fig. 3.1. Each module is organized in processes and functions, dedicated to LCU and/or WS applications, which use specific commands to perform all actions foreseen for the related subsystems control and management.

The TCS is subdivided in many modules, identified by the functional responsibility. A module consists of:

- One/more processes

- Database
- Command Interface

The typical features of VST software modules are:

- Every module runs on workstation and/or LCU platform.
- The tif module provides a single interface for external users. Such users do not see the modules structure and interact only with tif.
- The functional modules provide coordination features on the workstation side.
- The subsystem modules provide real-time functionalities on the LCUs.
- The TCS workstation contains the TCS Online database. Every functional module correspond to a branch in the database.
- Every subsystem has its own LCU database and a scan image on the TCS WS database.
  - The online database is used to have configurable any dependency between modules. For example:
  - A module is implemented as one or more processes.
- One of the processes (called control) provides the only interface to other modules and coordinates the other internal processes.
- A module defines a single database branch.
- The root point of the branch provide attributes for:
  - State
  - Substate
- Every process has its own private sub-branch.
- TCS modules are enough independent to be possible developing and testing one by one.
- A few unavoidable dependencies are due to the fact that some modules need services and information provided by other ones.
- Common services are provided through:
  - Libraries
  - Database classes

This defines the hierarchy between modules.

- The database used is a real-time distributed one, based on the hierarchical model, extending the classical RTAP database application, introducing Object Oriented concepts like inheritance, overloading, methods and class types.
- The design of every process consists mainly in the design of the independent objects providing the methods to be attached as callbacks to the event.
- The following events are handled:
  - Commands
  - Replies and error replies
  - Time out notifications
  - Cyclic timers
  - Database events
  - Alarms
  - Signals
  - File input
- The callbacks are stored in a dynamic dictionary. They must return within a "command response time" of 1 second.
- A wide set of ready-made classes implements standard behaviors and event handling protocols.
- A standard VST TCS module application must be able to provide standard public commands, (such as ping, init, state, status, standby, online, off, selftest, test, simulat, stopsim, verbose, version, stop, exit, kill, break), and standard database points.
- The commands handled by TCS are splitted in "immediate", "transfer", "multi step".
- The immediate commands are the simplest case: an application receive a command, deals with it and sends an immediate reply to the originator. They are implemented writing just one callback to receive the command.
- "transfer" commands are a typical case for workstation coordination applications: an application receives a command, deals with it and send one sub-command to the controlled processes. When they are done it must sends the final reply to the originator.
- "transfer" commands are implemented using the standard evhCommand methods and some support callbacks to receive the command and send the replies.
- In multi step an application receives a command, deals with it and send one sub-command to a controlled process. When the reply comes, some new elaboration is done and a new command is sent and so on for as many steps are needed.
- The "multi step" commands are implemented using the standard evhCommand methods.

4. Tracking and axis control module software description

The basic purpose of the tracking module is to drive the tracking axes of the telescope to follow the apparent motion of an astronomical object. Furthermore, the motion of a tracking axis to a new, fixed position is controlled by the tracking module, that can be considered as the main telescope subsystem to be described in more detail. The controlled axes are, respectively, azimuth axis, altitude axis, instrument rotator and adapter rotator axes. In fig. 4.1 the tracking functional diagram is shown. The tracking module is split in a real-time section and an administration and coordination section. The former is responsible for calculating position references with accurate timing, and the latter is doing coordination of tracking axes, setup file handling, command reception and checking and other less time-critical work. The real time part is implemented on LCUs, while the coordination part on WS. For the VST there will be only one WS dedicated to axes control, but the LCU tracking operations will be integrated into each axis LCU. The list of all LCUs currently involved in tracking is maintained by the Mode Switching module and only used by the tracking module, that computes also position references in a general way, that can be used for every tracking axis. The actual position control of an axis is device dependent and is not part of the tracking module. By this way, in each LCU of a tracking subsystem, the LCU part of the tracking module will co-exist with the application module that performs and handles position control and various auxiliary functions related to that. These modules are called ACM (Axis Control Modules). In particular, the complete, corrected reference angles calculated by the tracking module, are passed to the host ACM by call to a specific function, that has to be provided by the host ACM modules and adapted to the peculiarities of that particular axis. Then the ACM module monitors the position errors and, based on actual axis substate, it modifies the axis substate. When this event occurs, particularly from PRESETTING to TRACKING or IDLE states, it means that a previous presetting request is ready and the correct reply to WS tracking module can be sent. Other changes of substate will not cause any action by the tracking module. From the position limits and interlocks point of view, these signals are handled by the ACM module. The tracking module LCU part will be event triggered by these events, to allow for sending an error reply on an active command.
The ACM subsystem takes into account the need to create a generic software structure to support the individual differences between the Altitude/Azimuth axes, the Adapter/Rotator axes of both ESO VLT and NTT telescopes. Some additional elements will be introduced in this subsystem to support specific functional requirements of the VST. So, the basic purpose of the ACM is to drive one or more axis of the telescope to follow the apparent motion of an astronomical object. The ACM data flow diagram during tracking is shown in fig. 4.2. The coordinate transformation calculations are taken as samples for an internal interpolation of the position setpoint trajectory in time where the axis has to be positioned. In order to perform this task the ACM has to:

- Read the encoder system
- Read the tachometer system
- Read the UT time

- Switch on/off the drive
- Perform a preloading setup of motor torque
- Close a position loop
- Close a speed loop
- Provide the standard commands to control the system

Additionally the following secondary functions have to be performed:

- Monitoring and handling error conditions
- Monitoring and handling of interlocks
- Control of the encoder system, including start-up and calibration
- Control of the tachometer system
- Control of the preloading system
- Control of the drive subsystem and interfacing to all HW I/O
- Provide commands for tuning, test and maintenance

This LCU software module contains the control logic to position an axis using a position controller. It also provides a command interface for the standard commands necessary to start, stop and move an axis in tracking mode. Additionally it includes a set of maintenance functions which allows to preset and move the axis in speed mode. (without position control). Furthermore, the axis module provides a mechanism to define and configure signals for LCC and the database, the access to the encoder subsystem and to the drive subsystem, the interlocks, proximity limit switches, monitoring signals and other hardware components of the axis. The original module, provided by ESO, will be integrated with speed loop and preloading subsystems, needed specifically for VST TCS. The main features of the axis control module are:

- position controller with
- PI controller for errors
- Intermediate polynomial controller to achieve a smooth transition between the two used position control algorithms
- Velocity loop with:
  - PI controller for errors
  - Tachometer signal oversampling
- Adaptive motor torque preload

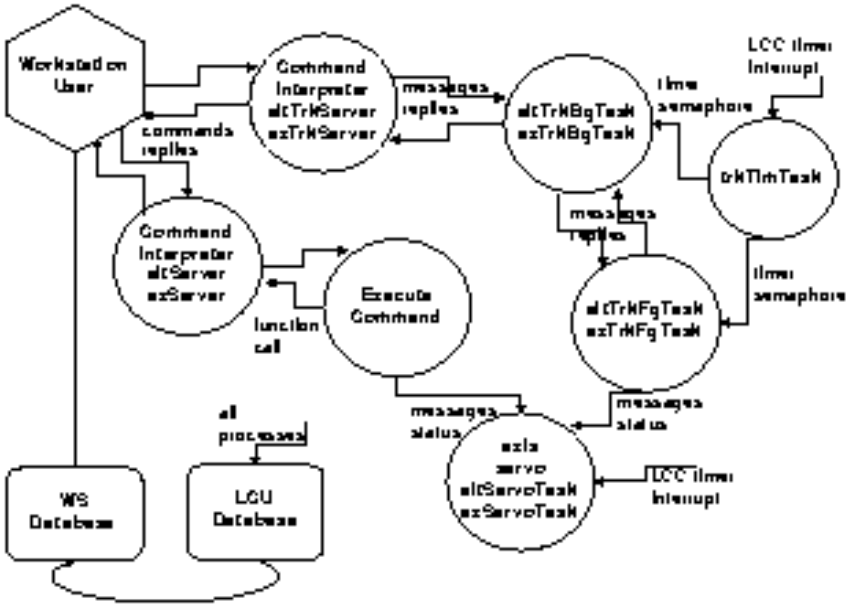

Fig 4.2 ACM data flow diagram during tracking

- A filter for the velocity setpoint limiting the velocity, acceleration and jerk
- A user definable servo sampling frequency up to 500 Hz
- A linear interpolation of position references received to the servo frequency
- Support of end limit interlocks and proximity switches
- Automatic deceleration and switch off of the drives in case of hardware/software errors
- Support of status database attributes
- Maintenance commands to preset and move in speed mode
- Maintenance commands to calibrate an encoder system
- Design of the servo task as a finite state automaton

5. Conclusions

The VST will follow the standard concepts adopted by ESO for VLT and the high quality experience acquired by TWG in several projects related to telescopes, (TNG, TT1) and instrumentation, (VIMOS, NIRMOS, GOHSS, DIMM). Doing so the global system coherence and reliability will increase and an easier maintenance will be guaranteed. The integration of the VST telescope in the ESO environments will be automatically ensured.

Acknowledgement
The authors wish to thanks Prof. G. Sedmak, VST Project Manager, Prof. M. Tarenghi, Prof. R. Kurz and Prof. G. Raffi of European Southern Observatory (ESO).

References

1 Technology Working Group, *VST Preliminary design supplement*, section 6, Issue 1.0, 08.04.99, OAC Naples
2 Technology Working Group, *VST Technical Proposal*, 06.10.97, OAC Naples
3 Technology Working Group, *VST Preliminary Design Review*, section 6, Issue 1.0, 15.01.99, OAC Naples