ATACAMA LARGE MILIMITER ARRAY

ALMA COMMON SOFTWARE

USM ACS Developers

Softaware Ingeniering
–
gTCS Design Manual
Design Manual

Doc. No. USM-MAN-0001

Issue 1.1

Date 03/08/2005

Prepared for Review

Keywords: ACS, UTFSM, gTCS

	Name	Date	Signature
Prepared	Nicolas Troncoso	03/08/2005	
Approved	Nicolas Troncoso	—/—/2005	
Released	Mauricio Araya	—/—/2005	

This page was intentionally left blank

Change Record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	03/08/05	All	First Draft.
1.1	07/09/05	All	Added State Diagram, Modified Usecases.

This page was intentionally left blank

Contents

This page was intentionally left blank

1 DRAFT-TODO

- check the crossed references
- check the tcs life cycle usecases
- Add the state diagram
- Add the subsystem diagram

2 INTRODUCTION

2.1 PURPOSE

The aim of this document is to detail the specifics of the gTCS design. User requirements, use cases, state diagrams, interaction components and interaction diagrams.

The intended audience are software developers.

2.2 SCOPE

This document describes and details the user requirements and logical construction of the gTCS. The audience of this document are the gTCS developers, or any one how wants to know the design specifications of the gTCS.

2.3 APPLICABLE DOCUMENTS

No Document are applicable at this moment.

2.4 ABBREVIATIONS AND ACRONYMS

The following abbreviations and acronyms are used in this document.
gTCS Generic Telescope Control System

3 OVERVIEW

The main objective of this document is to provide a clear design specification for the gTCS. To achieve such goal this document is divided in to parts; user requirements and design. User requirements specify the goal to be achieved by the gTCS. The design specifies the software model that will be implemented by the ACS UTFSM development team.

4 USER REQUIREMENTS

4.1 Specific gTCS Functional Requirements

4.1.1 Basic life-cycle control

A simple control of the basic lifecycle of the gTCS. It must include at least Start, Stop and Shutdown functions, and a log of events.

4.1.2 Display Telescope actual position and status

The user should know at every minute the general Telescope status, all the Axis information and localization/time information. At least, the actual Equatorial and Horizontal coordinates should be displayed as well.

4.1.3 Move the Telescope to a new position

Provide the presetting interface, that means to change the Telescope Position using Equatorial or Horizontal coordinates. Also, fixed presets like Zenith, Park, Sunset, etc must be implemented.

4.1.4 Compensate the Earth movement

Automatically follow the preseted coordinate compensating the Earth movement (tracking). An on/off control must be attached to this feature.

4.1.5 Additional velocity to tracking

If the tracking system is running, an additional velocity could be added to track near celestial objects.

4.1.6 Pointing Corrections

Directly modify the pointing model. The idea is to correct errors of tracking and presetting.

4.1.7 GUI interface

All previous requirements should be represented in a single panel, organized as the VLT panel. An alarm monitor should be included as well in this panel.

4.2 Project Functional Requirements

4.2.1 Connect the gTCS to a OpenGL model

An OpenGL model, simulating an Equatorial mount must be connected to the gTCS, simulating encoders and motors.

4.2.2 Connect the gTCS to Scarlet

A toy model of a Horizontal (Azimuth/Altitude) Telescope should be implemented with the RCX and the Lego, including motors and encoders. This toy model should be the final test of the gTCS before presetting our results.

4.3 Non-Functional Requirements

4.3.1 Provide a maintenance interface

Study if there is need of a maintenance interface. If not (ACS provides all the maintenance interface that the project needs), then document how to maintain the Telescope using only ACS tools.

4.3.2 The model should be extensible

The code should be extensible to implement others features like auto-guiding, mode switching or active optics.

5 DESIGN

5.1 SYSTEM FUNCTIONS

Ref.	Function	Category
gTCS life cycle		
R1.1	init gTCS services	Evident
R1.2	run gTCS services	Evident
R1.3	stop gTCS services	Evident
R1.4	kill gTCS services	Evident
R1.5	shutdown gTCS services	Evident
Data Acquisition		
R2.X	Geters	Hidden
Control		
R3.1	Move the motor X	Evident
R3.2	Read encoder	Hidden
Time System		
R4.1	Read Sideral time	Hidden
Pointing model		
R5.1	Calculate pointing corrections	Evident

5.2 STATE DIAGRAM

- Global States (modswitching):
  - OFF
  - Preseting
  - Observing
  - Error - Substates
  - Idle
  - Calibrating pointing
- Tracking:
  - On
  - Idle
  - Emergency Stop
  - Error
    - Software Limit
    - Inter Lock
    - Comunication
- Pointing:
  - Idle
  - Calibrating
  - Error
    - Software Limit

        - Comunication

- Preseting:
    - Idle
    - Preseting
    - Error
        - Software Limit
        - Comunication

Global States Condition:

Global State	Tracking State	Pointing State	Preseting State
off	off	off	off
Presting	on or idle	idle	preseting
Observing	on	idle	idle
Error	Any module in a error state		
Idle	idle	idle	idle
Calibrating	on	calibrating	idle or preseting

5.3 USE CASES

5.3.1 Initialize the gTCS service

- **Actors:** GUI, Command Line
- **Type:** Primary
- **Description:** The GUI, or Command invokes all initialization sequence for the gTCS.
- **Preconditions:** gTCS system is not loaded in memory.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI, or Command line runs the init procedure of the gTCS	2. The system starts up all subsystems and changes its status to stand-by state.

- Alternate flow of events:
  - **Line 2:** The system can not load, errors will be reported.
  - **Line 2:** One of the subsystems could not be loaded or is in an error state, error will be reported.

5.3.2 Run the gTCS service

- **Actors:** GUI, Command Line
- **Type:** Primary
- **Description:** The actor invokes the run sequence to change the status of the system from stand-by to operative.
- **Preconditions:** gTCS system is in stand-by state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI, or Command line execute the run procedure of the gTCS	2. The system changes to operate state.

- Alternate flow of events:
  - **Line 2:** The system can not switch to operative state and returns to stand-by indicating and error.
  - **Line 2:** The system returns to stand-by because the cancel routine was invoked.

5.3.3 Stop the gTCS service

- **Actors:** GUI, Command Line
- **Type:** Primary
- **Description:** The actor invokes the stop sequence, and the system changes status from operative to stand-by.
- **Preconditions:** The system must be in operative state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when the GUI or the command line invokes the stop sequence	2. The TCSi invokes the stop routine. When the system is stopped the system (modswitching) changes to stopped status.

- Alternate flow of events:
  - **Line 2:** One of the components does not respond to the stop sequences. The system changes to an error state, reporting the error.

5.3.4 shutdown the gTCS services

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:** The actor invokes the shutdown sequence and the system changes it status to off state.
- **Preconditions:** The system must be in stand-by state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI or command lines invokes the shutdown sequence	2. The system archives the pointing model, and all logs that have not been flushed out already. The system terminates all gTCS component execution.

- Alternate flow of events:
  - **Line 2:** One of the components does not shutdown. The system continues to shutdown all other components and changes to an error state. Reporting the error.

5.3.5 kill the gTCS services

- **Actors:** GUI, Command line
- **Type:** primary
- **Description:** The actor invokes the kill sequences and the system terminates all component execute in any possible fashion.

- **Preconditions:** none
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI or command line invokes the kill sequence	2. The system terminates all components in any possible way, changing it status to off (unloading it from memory). Before termination it will try to save the pointing model and all other logs.

5.3.6 Move the telescope

- **Actors:** Operator, GUI, Command line
- **Type:** Primary
- **Description:** The actor selects a type of movement.
- **Preconditions:** The system must be in operative state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when operator wishes to move the telescope to a new position. The operator must choose if he wishes to move the telescope doing a new preset or if he wishes to move the telescope to a fixed position.	2. The usecase continues in 5.3.8 if the movement is by preset. The usecase continues in 5.3.7 if new position is fixed.

- Alternate flow of events:
  - **Line 2:** Movement is not posible because the telescope is in an error state.

5.3.7 Move the telescope to a fixed predefined position

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:** Move the telescope to one of the following position; Zenith, Park, Sunset.
- **Preconditions:** The system must be in operative state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when the GUI or command line select a fixed predefined position	2. The system asks modswitching to chechk if a preseting operation is posible. And changes the status to: Preseting5.2 3. The system does a preseting using a predefined preset file (this file must turn the tracking system off).

- Alternate flow of events:
  - **Line 2:** Modswitching does not allow a movement operation, an error must be reported.

5.3.8 Move the telescope to a new position

- **Actors:** Operator, GUI, Command line
- **Type:** Primary
- **Description:** Move the telescope to a new position using a preset file.
- **Preconditions:** The system must be in operative state.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when the operator suplies a new preset file	2. The system asks modswitching to chechk if a preseting operation is posible. And changes the status to: Preseting5.2. 3. The system verifies the coordinates, checks if they are observable. 4. The system preseting subsystem sets a new position at the tracking subsystem. 5. When the new position has been reached, the preseting subsystem informs modswitching and changes the system status to Observing5.2.

- Alternate flow of events:
  - **Line 2:** Modswitching does not allow a movement operation, an error must be reported.
  - **Line 3:** The new coordinates are not valid. Preseting is cancelled and modswitching is set to: idle5.2

5.3.9 Change the tracking system velocity

- **Actors:** Operator, GUI, Command line
- **Type:** Primary
- **Crossed references:** R2.X
- **Description:** Modify the actual tracking speed.
- **Preconditions:** The system must be on stand-by or operative.
- **Normal flow of events:**

Actor's Action	System Response
1. This use case start when GUI or command line modifies the tracking speed	2. The system checks in modswitching if a speed change is posible. 3. The system modifies the tracking velocity.

- Alternate flow of events:
  - **Line 2:** The input speed is out of range, the system ignores the new tracking speed.

5.3.10 Calibrate telescope

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:**Calibrate the telescope.
- **Preconditions:**
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when the Operator calibrates the telescope	2. The system continues with 5.3.11 if the auto-calculate shift use case is selected, or continues with 5.3.13 the input new pointing model usecase is selected, or continues with 5.3.12 if the calculate pointing model usecase is selected.

5.3.11 Auto-calculate shift

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:** Calculate the shifts between the aimed position and the real position of a set of stars.
- **Preconditions:**
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI selects Auto-calculate shift	2. The system reads from the archive the set of stars to be aimed. 3. For each star the system stores the shifts in the archive.

5.3.12 Calculate Pointing Model

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:** Calculate a new pointing model from the shifts stored in the archive.
- **Preconditions:**
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI selects Calculate Pointing Model	2. The system reads from the archive the set of shifts. 3. The system calculates a new pointing model and stores it in the Archive.

5.3.13 Input new pointing model

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:** Input a new pointing model into the pointing subsystem.
- **Preconditions:**
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI selects input new pointing model	2. The system reads from the archive the new pointing model.

5.3.14 Adjust aiming

- **Actors:** GUI, Command line
- **Type:** Primary
- **Description:**Move the telescope in individual steps to adjust the actual position. This modification does not affect the coordinates, this means that it is an implicit pointing model correction.
- **Preconditions:**
- **Normal flow of events:**

Actor's Action	System Response
1. This use case starts when GUI or command line sets a pointing correction including direction and number of steps	2. The system tell the tracking subsystem to step in a direction.

___oOo___