Goal-oriented architecture for telescope control software

L. Andolfato *[a], R. Karban[b]
[a]European Southern Observatory, Karl-Schwarzschild-Strasse 2, Garching bei München, Germany;
[b]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA USA

ABSTRACT

Modern telescope control systems control a multitude of sensors and actuators physically distributed across subsystems. Traditionally, control systems are specified to compute system outputs based on reference setpoints despite environmental or internal disturbances. The control strategies are typically tightly coupled and very prescriptive, defining how exactly the control is to be achieved.

In a goal-oriented approach, the control system architecture emphasizes the operational aspect, specifying the behavior at a higher level in terms of operational goals, considering several aspects such as environmental conditions and required wavefront quality. The result is an architecture that can more autonomously handle different control strategies, taking into account systematically the physical states of the system which are relevant for control.

This paper demonstrates the feasibility of a goal-oriented approach by analyzing four fundamental functions: pointing, tracking, offsetting, and dome vignetting. The resulting SysML analysis and design models are used to develop a software prototype for distributed telescope control. Typical operational scenarios are discussed and compared to the ESO Very Large Telescope control software architecture.

Keywords: Telescope Control Software, Goal Oriented Control Software, State Analysis, SysML, Control Software Architecture

1. INTRODUCTION

Astronomical observations are characterized by a challenging environment made of laws of physics, rules defined by the observatory and society, and environmental conditions. For example, a telescope can be deformed by gravity and the trajectory of a celestial object as observed from earth is described by math. At the same time, it is forbidden by international laws to point a laser at an aircraft and it is hazardous for an observatory to point the telescope primary mirror to the sun. Moreover, ground based observations are affected by weather and atmospheric conditions (Table 1).

Table 1. Environmental challenges in telescope control.

Challenges	Affect	Examples
Physics	What is observed Telescope	How stars move across the sky. Telescope structure or mirror deformations.
Rules	Telescope Humans	Don’t point to the sun. Don’t point the laser to aircrafts.
Environmental Conditions	Telescope Atmosphere	Mirror condensation. Visibility.

*landolfa@eso.org; phone ++49 89 3200 6796; eso.org

Traditionally, telescopes systems are built on an architecture that focuses on specific aspects of a single observation such as: target selection, calibration, pointing, tracking, guiding, active optics, adaptive optics, target data acquisition, target data preprocessing. Instead, a goal-oriented architecture considers the combination of all these aspects together allowing for an optimized end-to-end system supervision and control. This approach can be extended to cover the sequence of observations in one night or even in the whole observation period (which can last for months). The idea is that by increasing the amount of information that can be used by the control software, it should be possible to increase the amount of autonomous decisions and therefore limit the need for human intervention.

New generation telescopes are larger, with many more components (from 1000s to 10000s I/O points) and features (integrated adaptive optics, laser guide stars, large segmented mirrors, etc.). This increase in size results in a higher probability of failures. In order to guarantee high system reliability, the control system needs to be able to quickly adapt to failures. Such adaptations are currently performed manually by reconfiguring the system but, because of the size of the new systems, the operator may not be able to cope with the amount of information produced (measurements) and required (configuration).

This paper describes the results of a prototype developed to investigate how to adopt a goal-oriented architecture for the development of a subset of telescope functionalities: the main structure and dome. The paper is structured as follows: chapter 2 introduces the goal-oriented control applied to the telescope domain using the design of the prototype as running example; chapter 3 describes the implementation of the prototype; chapter 4 summarizes the lessons learned from the prototype, and chapter 5 proposes a possible reference architecture for goal-oriented telescope control that could be investigated in future work.

2. GOAL ORIENTED CONTROL

2.1 Overview

Navigation systems such as Google Maps are very successful services which coordinate millions of cars in a loosely coupled fashion to reach their destinations. These services provide a good analogy of goal-oriented control systems. When a person wants to get from point A to point B, the navigation system will plan a route based on the car's location, certain constraints (e.g. available roads), and information about the environment (e.g. traffic situation along the route). The user can provide additional constraints such as avoiding toll roads or choosing the kind of vehicle. After the requirement is specified, the navigation system provides a sequence of instructions (Goals) which are valid for a certain position, for example "Keep straight for the next 10 miles and then turn right".

In goal-oriented control, the navigation system plays the role of the Goal Planner since it prepares the sequence of Goals that the driver, who acts as a Control System, has to execute by monitoring and sending commands to the car, the System Under Control (SUC). The driver will try to do its best to follow those directions by monitoring the environment and the car, and also by interacting with the car. The person observes the car's speed, position, fuel level, the traffic lights, and exercises control by changing the position of the pedal or by simply pushing buttons on the steering wheel. The information monitored by the driver are called State Variables (SV): they represent attributes of the SUC (such as speed, position, fuel level) or of the environment (traffic light). SVs can affect each other, for example when changing the accelerator position the fuel level will change. These cause-effect relations can be captured in a model, the State Effect Model (SEM).

The navigation system not only provides the Goals but also monitors them and, if the driver fails to achieve one, it will re-plan the route and schedule new Goals. Note that the navigation system supervises "what" has to be achieved, not "how", which is completely left to the driver.

In goal-oriented control, the Control System is made up of Controllers, Estimators, and SEMs. The Controllers consult the SVs and send commands to the SUC to achieve the assigned Goals. Estimators estimate the SVs by processing measurements from the SUC and by consulting other SVs. Communication to the SUC is facilitated by the Hardware Adapters which abstract the details of the underlying hardware. The SUC and the environment represent the "Real World". The SEMs represent the Model of the Real World. Those four elements: Model, Real World, Controller, and Estimator are referred to as the Control Diamond and constitute the fundamental architectural pattern of goal oriented control[1].

The presented goal-oriented approach is an evolution of the State Based Control Architecture[2,3] that should help to achieve the following objectives:

- Capture an executable set of requirements expressed as Goals

- Provide guiding principles for control software design to implement effectively the control strategies devised by the systems engineers using an end-to-end system model
- Integrate control strategies and operations
- Enable traceability from operational scenarios to component behavior models to code
- Perform analysis of operational scenarios and component behavior
- Facilitate end-to-end system testing and tuning

The following sections describe the main concepts of the goal-oriented control by applying them to the design of a prototype implementing the control software for some of the typical telescope components: the main structure, including azimuth and altitude axes, and the dome, protecting the main structure. The prototype design is captured in a SysML model which specifies Goals, Procedures, Scenarios, State Variables, State Effects, Goal Monitor, Controllers, and Estimators.

2.2 Goals, Procedures, Scenarios, and State Variables

Goals are specifications of desired behaviors derived from the analysis of the requirements. They are statically specified during the design and include their constraints (e.g. duration). At design time their constraints can be "parameterized", with parameters to be specified at run-time. Figure 1 shows how the "Track on Target" Goal satisfies the two requirements related to pointing/tracking and dome rotation. Goals are composed into Scenarios which provide an exhaustive description of the intended system behavior and can consist of many interconnected Goals.

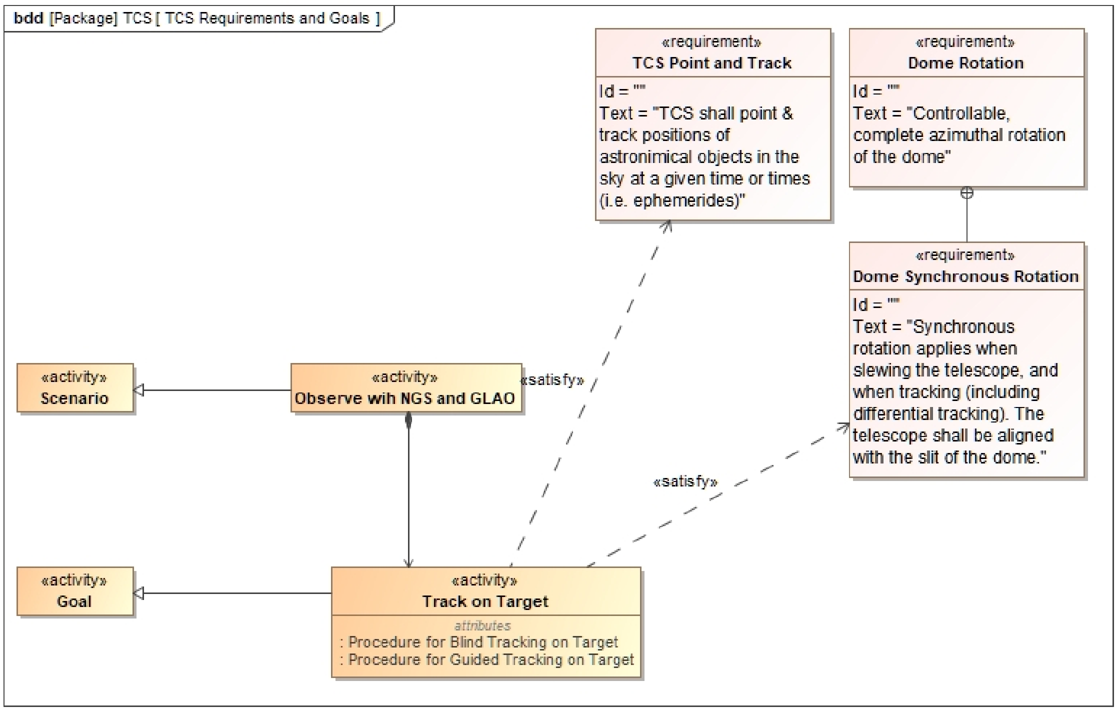

bdd [Package] TCS [TCS Requirements and Goals]
«requirement»
TCS Point and Track
Id = ""
Text = "TCS shall point & track positions of astronimical objects in the sky at a given time or times (i.e. ephemerides)"
«requirement»
Dome Rotation
Id = ""
Text = "Controllable, complete azimuthal rotation of the dome"
«requirement»
Dome Synchronous Rotation
Id = ""
Text = "Synchronous rotation applies when slewing the telescope, and when tracking (including differential tracking). The telescope shall be aligned with the slit of the dome."
«activity»
Scenario
«activity»
Observe wih NGS and GLAO
«satisfy»
«satisfy»
«activity»
Goal
«activity»
Track on Target
attributes
: Procedure for Blind Tracking on Target
: Procedure for Guided Tracking on Target

Figure 1. The "Track on Target" Goal and the "Observe with NGS and GLAO" scenario satisfy the system requirements. The "Track on target" goal can be achieved via the "Blind Tracking on Target" or the "Guided Tracking on Target" Procedure. Goals and Scenarios are represented in SysML as Activities.

The different ways a Goal can be achieved are specified using Procedures. A Procedure is basically a recipe made of a sequence of simpler Goals for accomplishing a high-level Goal. It can take input parameters and/or may depend on State Variables which represents "some" estimated knowledge of the system to control or of its environment (Figure 2). The definition of a Procedure helps in identifying what type of estimations/measurements (i.e. State Variables) have to be performed to achieve a Goal.

Scenarios, Goals, and Procedures are the building blocks to describe, at different levels of abstraction, "what" the system has to do. For example, one of the Goals in the "Observe with NGS and GLAO" Scenario is the "Track on target" which can be achieved either via the "Blind Tracking on Target" or the "Guided Tracking on Target" Procedure (Figure 3). Which Procedure to select, may be driven by the required tracking accuracy.

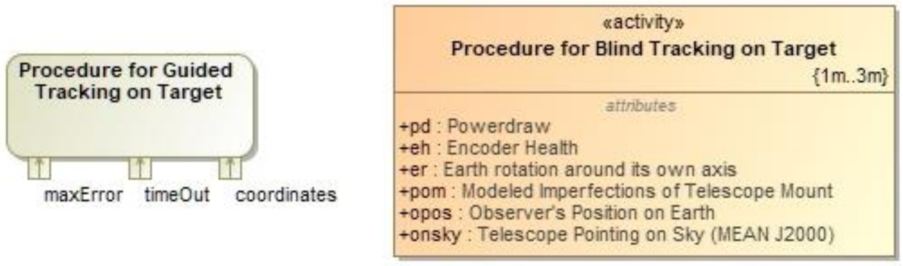

Procedure for Guided Tracking on Target
maxError
timeOut
coordinates
«activity»
Procedure for Blind Tracking on Target
{1m..3m}
attributes
+pd : Powerdraw
+eh : Encoder Health
+er : Earth rotation around its own axis
+pom : Modeled Imperfections of Telescope Mount
+opos : Observer's Position on Earth
+onsky : Telescope Pointing on Sky (MEAN J2000)

Figure 2. Procedures with parameters (Procedure for Guided Tracking on Target) and with State Variables (Procedure for Blind Tracking on Target). In SysML, a Procedure is modeled as an Activity.

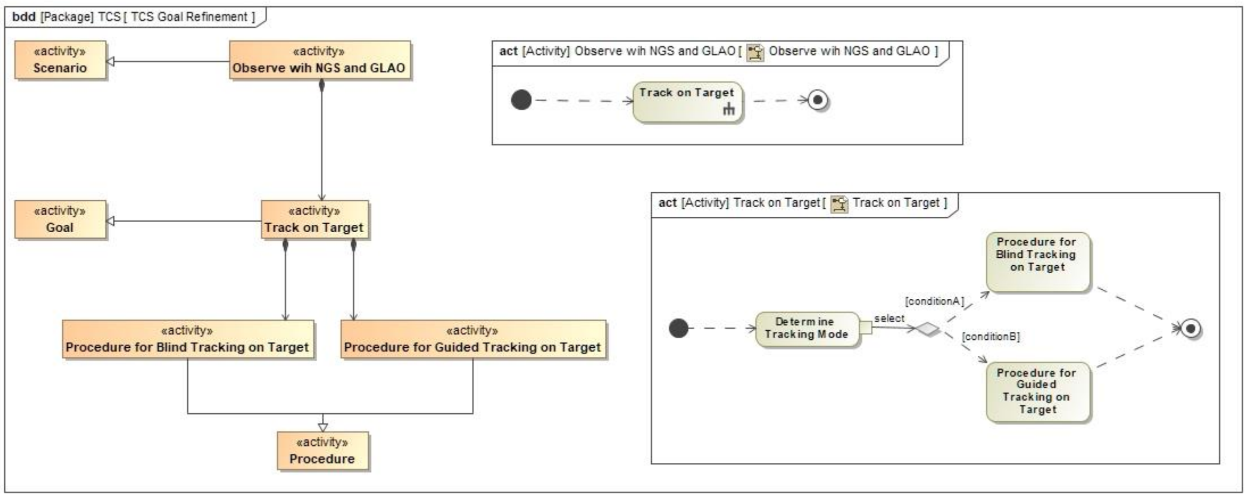

bdd [Package] TCS [TCS Goal Refinement]
«activity» Scenario
«activity» Observe wih NGS and GLAO
«activity» Goal
«activity» Track on Target
«activity» Procedure for Blind Tracking on Target
«activity» Procedure for Guided Tracking on Target
«activity» Procedure
act [Activity] Observe wih NGS and GLAO [Observe wih NGS and GLAO]
Track on Target
act [Activity] Track on Target [Track on Target]
Determine Tracking Mode
select
[conditionA]
[conditionB]
Procedure for Blind Tracking on Target
Procedure for Guided Tracking on Target

Figure 3. Scenarios, Goals, and Procedures can be used to specify the intended behavior of the system.

The main functional requirements for the prototype are summarized in Table 2. They are related to the kinematic functions of Main Structure and Dome and they are provided in terms of Goal specifications.

Table 2. Prototype high level requirements.

High Level Requirements	Goals	Specification	Associated State Variable
Pointing on target.	TransitionToTrackOnTarget	RA, DEC, Max Error, Time-out	Telescope Pointing On Sky
Tracking on target.	TrackingOnTarget	RA, DEC, Max Error, Time-out	Telescope Pointing On Sky
Avoid vigneting.	FreeSight	Max difference between Azimuth and Dome position, Time-out	Dome Tel Vignetting

Apply telescope offsets	If the intent is to re-point the Telescope: TrackingOnTarget If the intent is to improve pointing, additional evidence (e.g. error vectors from detector) should be used to improve the estimation of the Pointing State Variable.	RA, DEC, Max Error, Time-out	Telescope Pointing On Sky

Additional low-level requirements, like the possibility to move telescope axes individually to a relative or absolute position or to apply constant speed to a motor, have been implemented in the prototype although not reported here.

2.3 Goal Planner, Scheduler, Executor, and Monitor

The Goals in the Scenarios are processed by the Goal Planner which creates a list of basic Goals (Goal Network) obtained by selecting the most applicable Procedure according to the system state, environmental conditions, scheduling requirements, etc. The process applied by the Goal Planner to produce the Goal Network is called Goal Refinement. Goal Refinement can be done manually upfront during design or at run-time during operation.

The Goal Network is used by the Goal Scheduler to merge time overlapping Goals. The output of the Goal Scheduler is the sequence of Goals to be executed at a given time by the Goal Executor.

The Goal Monitor monitors whether a Goal is achieved or not. In case of failure, it may trigger a recovery action by propagating the failure to the Goal Planner. The Goal Planner can refine the Goal Network again by selecting a different Procedure.

2.4 State Effect Model

The State Effect Model is a model of the cause-effects relations between all SVs. Figure 4 shows a diagram of the SEM of the telescope environment considered in the prototype. The dashed arrow lines express the effect relationship. With the assumption that the science target and the guide stars (GS) are located close by, it is possible to derive that their wavefront ("Science Wavefront before the Dome" SV and "GS k Wavefront before the Dome" SV) are affected by the same atmospheric turbulences ("Atmosphere outside dome" SV) and earth rotation ("Earth rotation" SV). Moreover, since they are observed by the same telescope, they are also affected by the same observer position ("Observer's Position on Earth" SV). Note that in case the target and GS are far apart, dedicated environmental SVs would have to be added in the model.

Figure 5 instead shows a diagram of the SEM related to the telescope pointing. The "Observer's Position on Earth (Lat/Long)" affects the "Telescope Pointing on Sky". Note that the SVs should always be defined within a frame of reference (e.g. MEAN J2000). Similarly, the "Azimuth Position and Velocity" affects the "True Telescope Pointing on Sky (Alt/Az)" SV.

The prototype has to control the Dome, made of doors and wind-screens, to make sure that it does not obstruct the telescope field of view. The diagram of Figure 6 shows how the "Dome Tel Vignetting" SV is affected by the altitude and azimuth position and the doors position SVs.

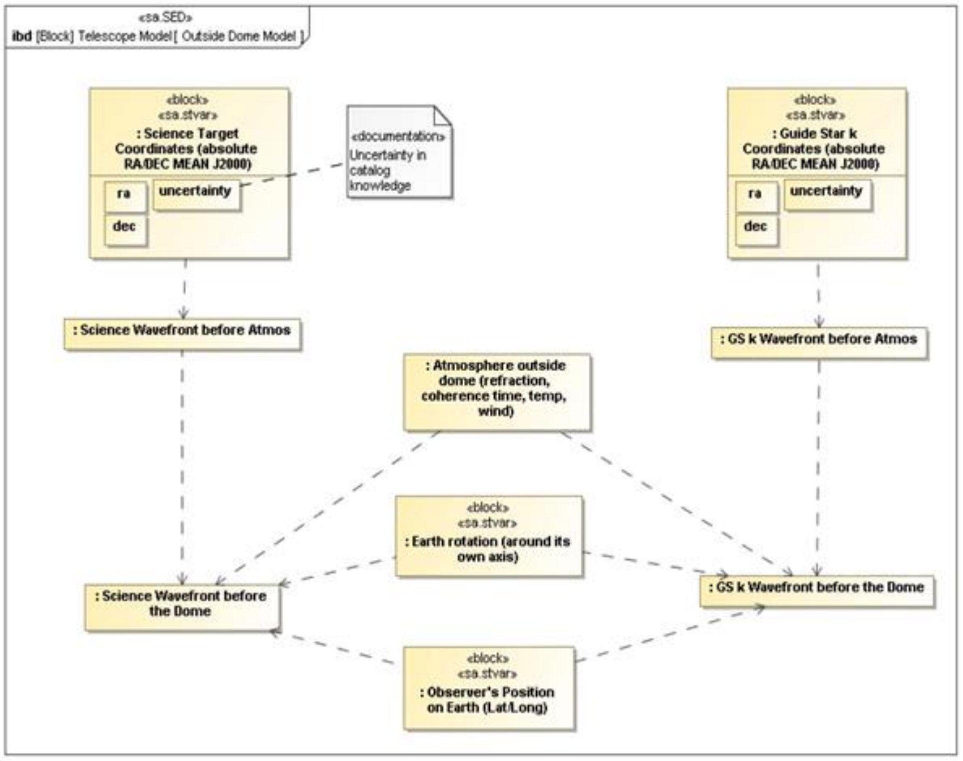

«sa.SED»
ibd [Block] Telescope Model [Outside Dome Model]
«block» «sa.stvar» : Science Target Coordinates (absolute RA/DEC MEAN J2000)
ra
dec
uncertainty
«documentation» Uncertainty in catalog knowledge
«block» «sa.stvar» : Guide Star k Coordinates (absolute RA/DEC MEAN J2000)
ra
dec
uncertainty
: Science Wavefront before Atmos
: GS k Wavefront before Atmos
: Atmosphere outside dome (refraction, coherence time, temp, wind)
«block» «sa.stvar» : Earth rotation (around its own axis)
: Science Wavefront before the Dome
: GS k Wavefront before the Dome
«block» «sa.stvar» : Observer's Position on Earth (Lat/Long)

Figure 4. State Effect Model of the environment outside the telescope. The stereotypes are loosely based on the concepts described in "An ontology for State Analysis: Formalizing the mapping to SysML"[4]: <<sa.stdvar>> is used for State Variables which can be controlled, <<sa.drvdSV>> is used for State Variables derived from other State Variables.

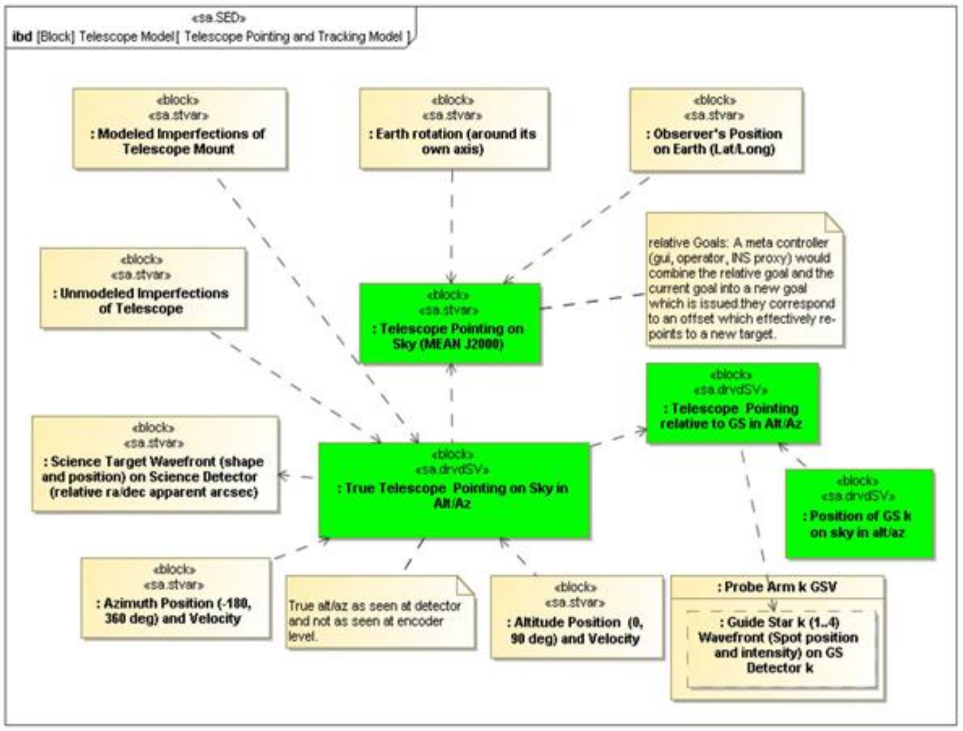

«sa.SED»
ibd [Block] Telescope Model [Telescope Pointing and Tracking Model]
«block» «sa.stvar» : Modeled Imperfections of Telescope Mount
«block» «sa.stvar» : Earth rotation (around its own axis)
«block» «sa.stvar» : Observer's Position on Earth (Lat/Long)
«block» «sa.stvar» : Unmodeled Imperfections of Telescope
«block» «sa.stvar» : Telescope Pointing on Sky (MEAN J2000)
relative Goals: A meta controller (gui, operator, INS proxy) would combine the relative goal and the current goal into a new goal which is issued.they correspond to an offset which effectively re-points to a new target.
«block» «sa.drvdSV» : Telescope Pointing relative to GS in Alt/Az
«block» «sa.stvar» : Science Target Wavefront (shape and position) on Science Detector (relative ra/dec apparent arcsec)
«block» «sa.drvdSV» : True Telescope Pointing on Sky in Alt/Az
«block» «sa.drvdSV» : Position of GS k on sky in alt/az
«block» «sa.stvar» : Azimuth Position (-180, 360 deg) and Velocity
True alt/az as seen at detector and not as seen at encoder level.
«block» «sa.stvar» : Altitude Position (0, 90 deg) and Velocity
: Probe Arm k GSV
: Guide Star k (1..4) Wavefront (Spot position and intensity) on GS Detector k

Figure 5. State Effects Model for telescope pointing. In green are the State Variables derived from other State Variables.

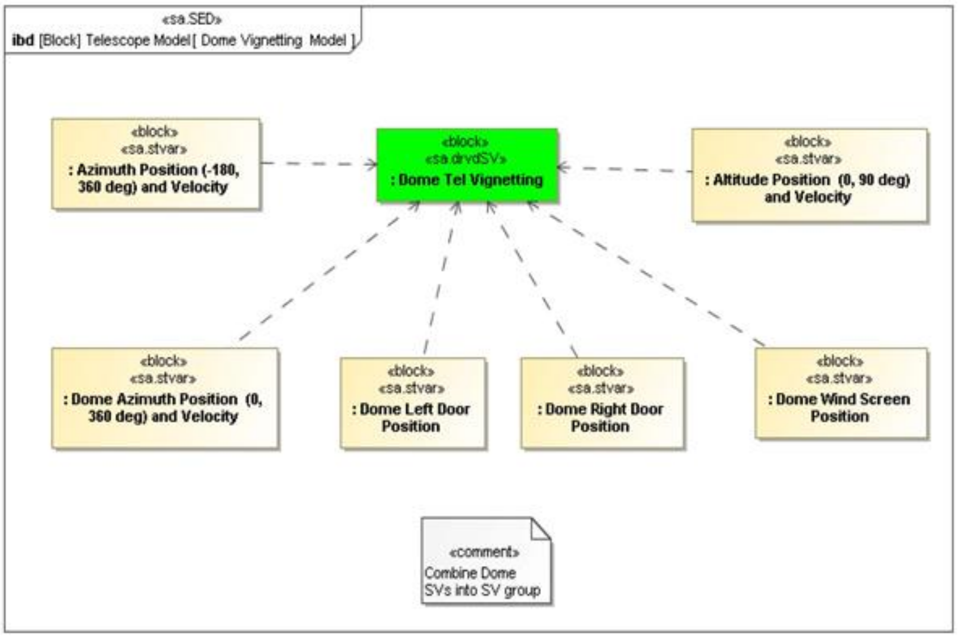

«sa.SED»
ibd [Block] Telescope Model[Dome Vignetting Model]
«block» «sa.stvar» : Azimuth Position (-180, 360 deg) and Velocity
«block» «sa.drvdSV» : Dome Tel Vignetting
«block» «sa.stvar» : Altitude Position (0, 90 deg) and Velocity
«block» «sa.stvar» : Dome Azimuth Position (0, 360 deg) and Velocity
«block» «sa.stvar» : Dome Left Door Position
«block» «sa.stvar» : Dome Right Door Position
«block» «sa.stvar» : Dome Wind Screen Position
«comment» Combine Dome SVs into SV group

Figure 6. State Effect Model for the Dome vignetting.

SVs are initially identified by analyzing the SUC. The number of SVs drives the size of the SEM and, in turn, of the control SW. Sometimes it is convenient to merge them to reduce the number of controllers/estimators or for performance reasons. In other occasions it may be more convenient for testability and reliability to keep them separated. SVs which cannot be estimated or controlled directly by the Control System, can be removed from the SEM although they have to be linked to the remaining SVs consistently.

Since SVs are constrained by Goals, there should be a correspondence between the list of Goals and the list of SVs.

2.5 Control System and System Under Control

Figure 7 shows the observatory structure considered for the prototype. The Telescope Control System (TCS) is part of the telescope as well as the Main Structure and the Dome (the SUC). The TCS uses the SVs identified in the SEMs (e.g. “Observer’s Position on Earth”, “Telescope Point on Sky”, etc.). Some of these SVs can be considered part of a Local Control System (e.g. “Az Encoder Health”).

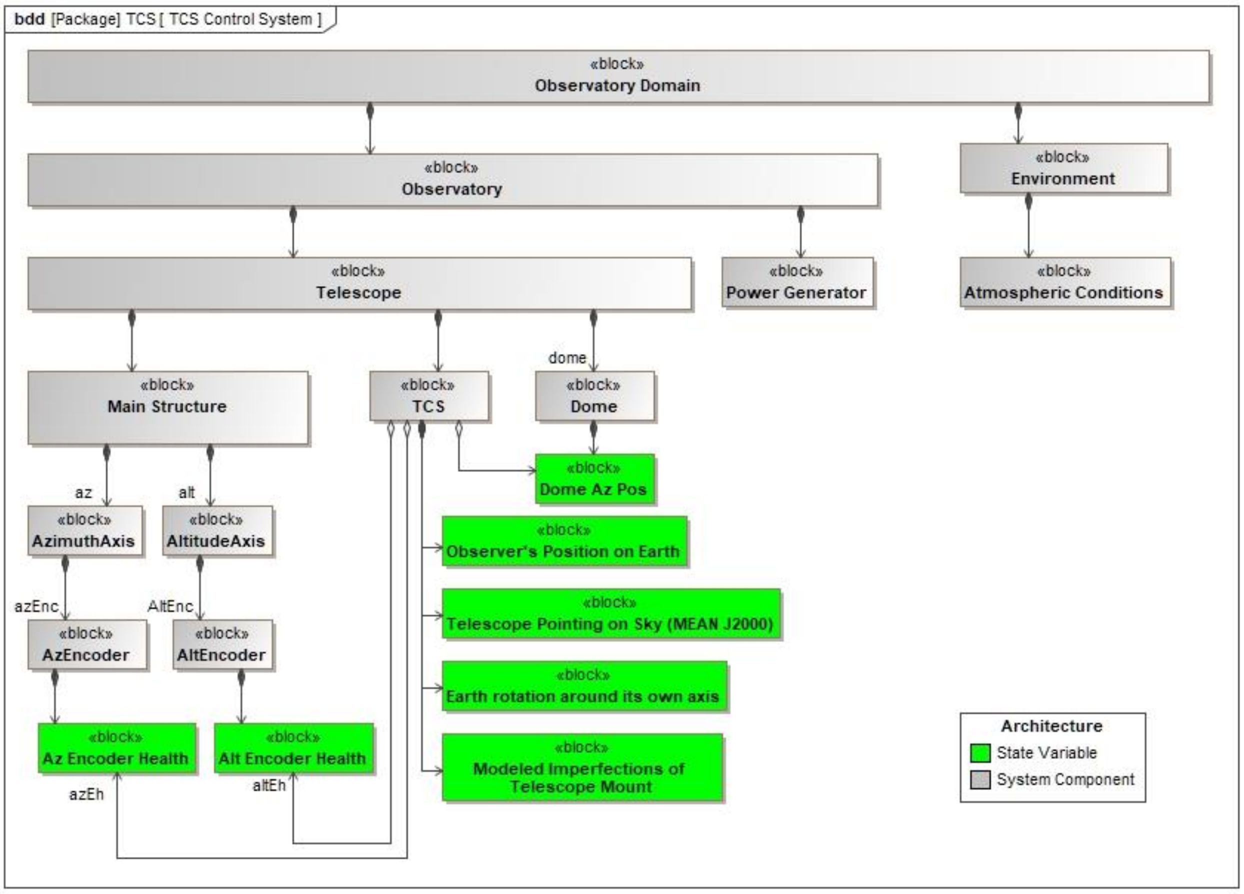

bdd [Package] TCS [TCS Control System]
«block» Observatory Domain
«block» Observatory
«block» Environment
«block» Telescope
«block» Power Generator
«block» Atmospheric Conditions
«block» Main Structure
«block» TCS
dome
«block» Dome
«block» Dome Az Pos
az
alt
«block» AzimuthAxis
«block» AltitudeAxis
«block» Observer's Position on Earth
«block» Telescope Pointing on Sky (MEAN J2000)
azEnc
AltEnc
«block» AzEncoder
«block» AltEncoder
«block» Earth rotation around its own axis
«block» Az Encoder Health
«block» Alt Encoder Health
«block» Modeled Imperfections of Telescope Mount
azEh
altEh
Architecture
🟩 State Variable
⬜ System Component

Figure 7. The Telescope Control System (TCS) and the System Under Control (Main Structure, Dome). The Telescope Control System maintains a list of State Variables describing the state of the System Under Control.

The control diamond pattern (Model, Real World, Controller, and Estimator, see section 2.1) is reflected in the breakdown of the control system as shown in Figure 8 where the Estimator uses Measurements from the SUC to estimate the SV, and the Controller issues commands to achieve a certain goal, e.g. “Avoid Vignetting”. The TCS is composed of as a set of Controller and Estimator structures.

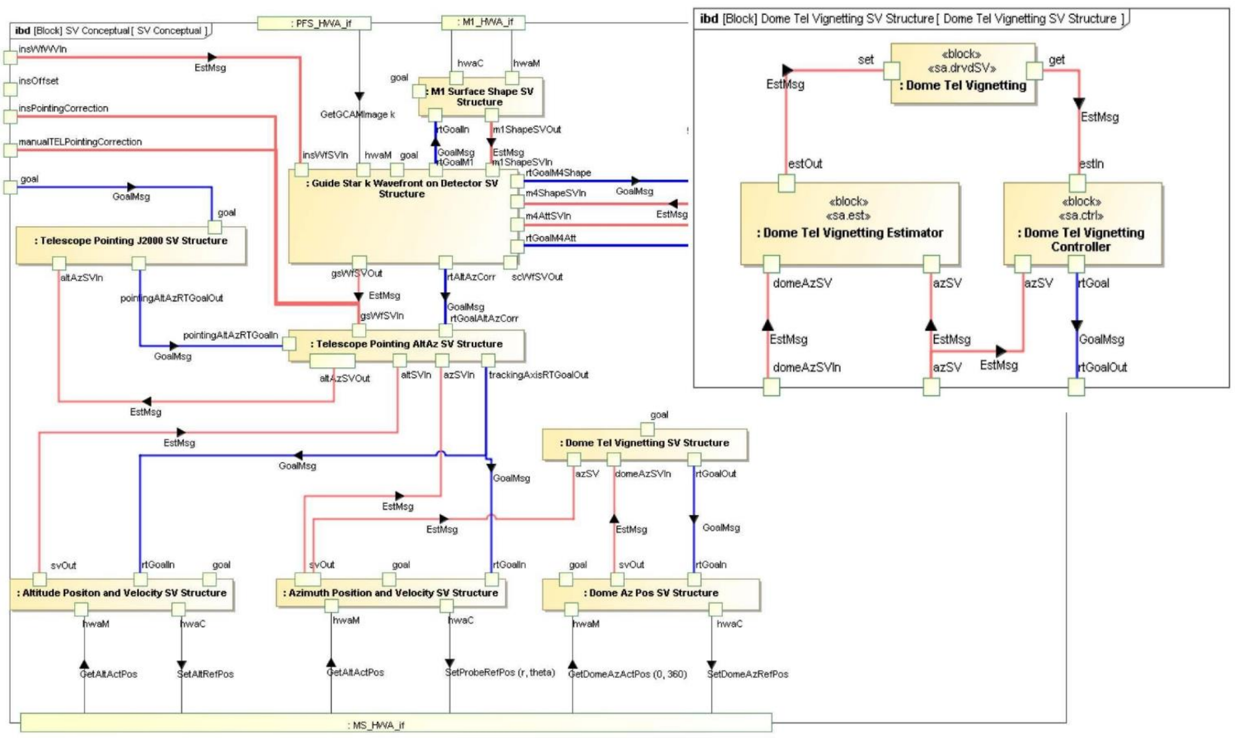
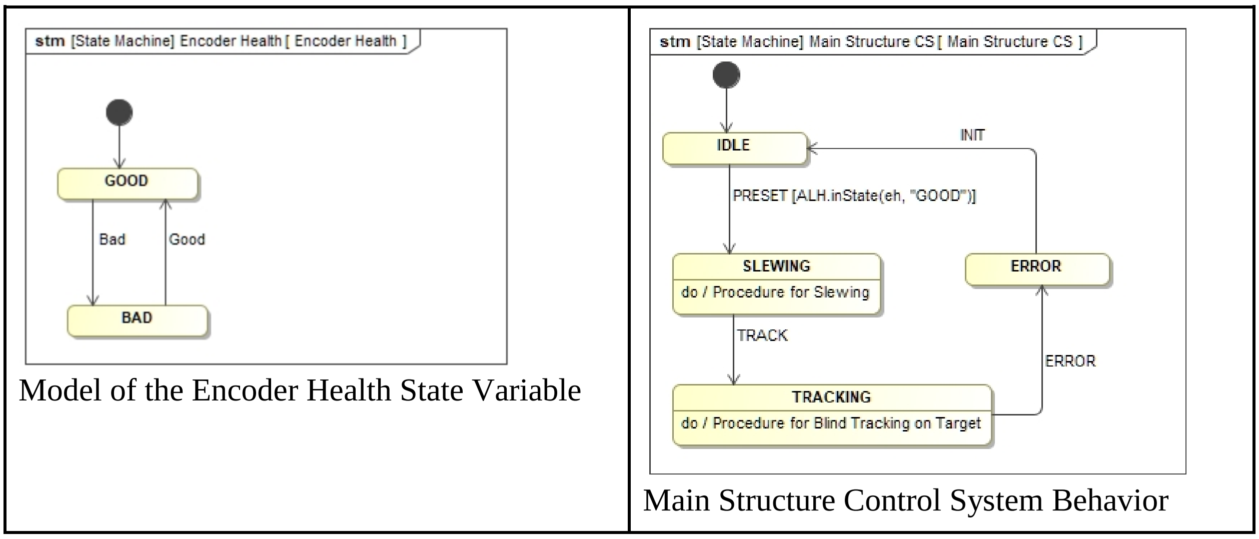

ibd [Block] SV Conceptual [SV Conceptual]
: PFS_HWA_if
: M1_HWA_if
insWfWVIn
insOffset
insPointingCorrection
manualTELPointingCorrection
goal
EstMsg
GoalMsg
: M1 Surface Shape SV Structure
hwaC
hwaM
GetGCAMImage k
rtGoalIn
m1ShapeSVOut
: Guide Star k Wavefront on Detector SV Structure
insWfSVIn
rtGoalM4Shape
m4ShapeSVIn
m4AttSVIn
rtGoalM4Att
: Telescope Pointing J2000 SV Structure
altAzSVIn
pointingAltAzRTGoalOut
pointingAltAzRTGoalIn
gsWfSVOut
rtAltAzCorr
scWfSVOut
gsWfSVIn
rtGoalAltAzCorr
: Telescope Pointing AltAz SV Structure
altAzSVOut
altSVIn
azSVIn
trackingAxisRTGoalOut
: Dome Tel Vignetting SV Structure
azSV
domeAzSVIn
rtGoalOut
svOut
rtGoalIn
: Altitude Positon and Velocity SV Structure
: Azimuth Position and Velocity SV Structure
: Dome Az Pos SV Structure
hwaM
hwaC
GetAltActPos
SetAltRefPos
SetProbeRefPos (r, theta)
GetDomeAzActPos (0, 360)
SetDomeAzRefPos
: MS_HWA_if
ibd [Block] Dome Tel Vignetting SV Structure [Dome Tel Vignetting SV Structure]
«block» «sa.drvdSV» : Dome Tel Vignetting
set
get
estOut
estIn
«block» «sa.est» : Dome Tel Vignetting Estimator
«block» «sa.ctrl» : Dome Tel Vignetting Controller
domeAzSV
azSV
rtGoal
domeAzSVIn
rtGoalOut

Figure 8. TCS conceptual architecture and Control Diamond pattern for Dome Vignetting control.

The behavior of Controllers and discrete SVs can be specified with state machines (SM). Figure 9 shows the SM of the Controller in “Telescope Pointing J2000 SV Structure” which uses the state of the Encoder Health SV to transition to “SLEWING” only if the encoder health is “GOOD”. Since the Controller is responsible to achieve a goal, it uses the Procedures to perform those actions in its states, such as “Procedure for Blind Tracking on Target’ in the state “TRACKING”. The example state machine shows only one procedure uses. Different procedures can be handled by different states or wrapped within a new procedure of a state.


stm [State Machine] Encoder Health [Encoder Health]
GOOD
Bad
Good
BAD
stm [State Machine] Main Structure CS [Main Structure CS]
IDLE
INIT
PRESET [ALH.inState(eh, "GOOD")]
SLEWING
do / Procedure for Slewing
ERROR
TRACK
ERROR
TRACKING
do / Procedure for Blind Tracking on Target

Model of the Encoder Health State Variable

Main Structure Control System Behavior

Figure 9. State Variable models and component behavior.

Figure 10 shows a sequence diagram of the TCS and related SVs generated by simulating INIT and PRESET commands.

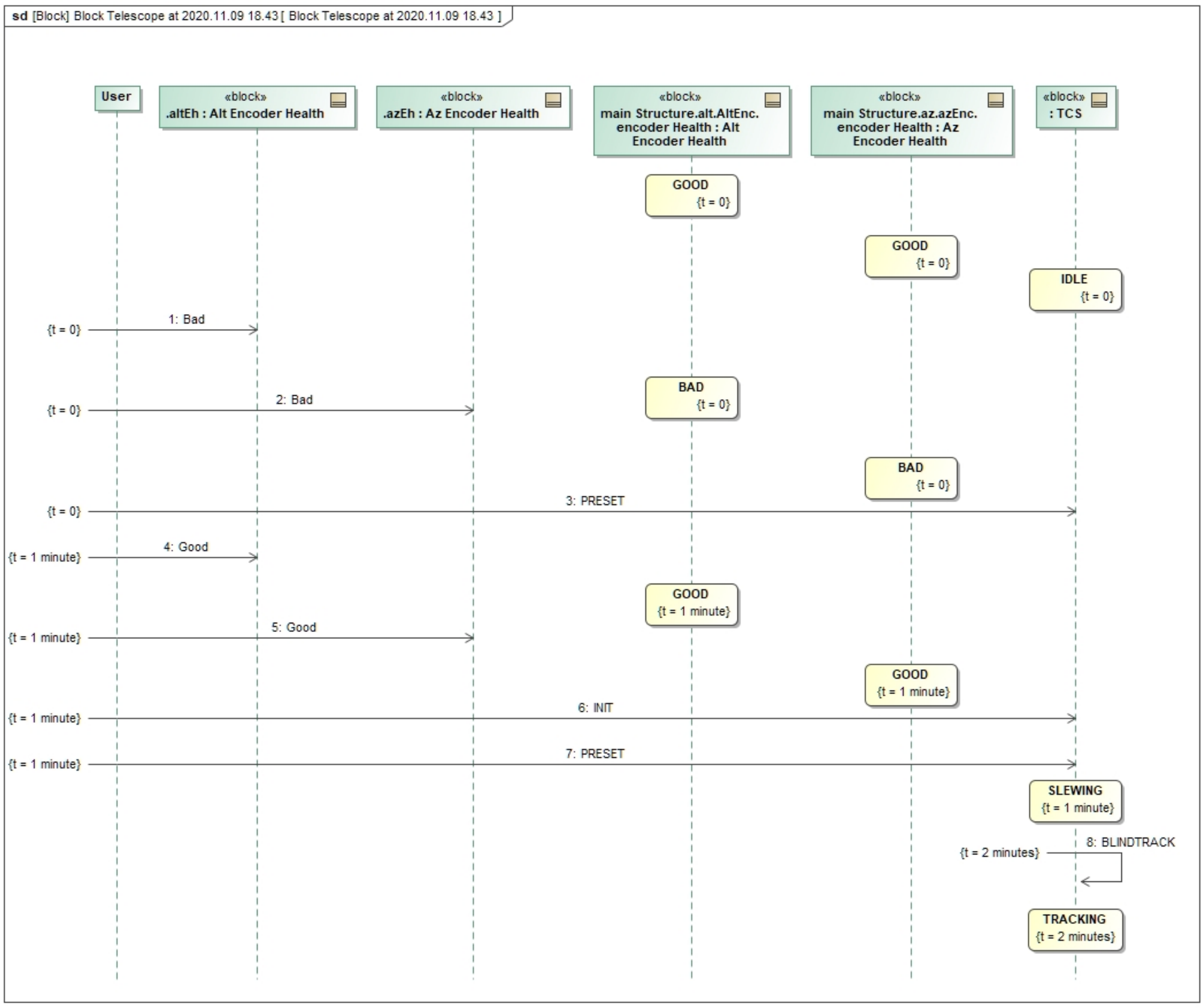

sd [Block] Block Telescope at 2020.11.09 18.43 [Block Telescope at 2020.11.09 18.43]
User
«block» .altEh : Alt Encoder Health
«block» .azEh : Az Encoder Health
«block» main Structure.alt.AltEnc. encoder Health : Alt Encoder Health
«block» main Structure.az.azEnc. encoder Health : Az Encoder Health
«block» : TCS
GOOD {t = 0}
GOOD {t = 0}
IDLE {t = 0}
{t = 0} 1: Bad
{t = 0} 2: Bad
BAD {t = 0}
BAD {t = 0}
{t = 0} 3: PRESET
{t = 1 minute} 4: Good
GOOD {t = 1 minute}
{t = 1 minute} 5: Good
GOOD {t = 1 minute}
{t = 1 minute} 6: INIT
{t = 1 minute} 7: PRESET
SLEWING {t = 1 minute}
{t = 2 minutes} 8: BLINDTRACK
TRACKING {t = 2 minutes}

Figure 10. Generated sequence diagram from scenario simulation.

2.6 Summary of Goal Oriented Concepts

Table 3 explains the concepts of a goal-oriented architecture. Note, that in an implementation they can be merged for optimization or implementation reasons.

Table 3. Goal oriented terminology and concepts

Concept	Description	Example
Command	Directive to an actuator to change the condition of one or more states in the System Under Control.	Change position of accelerator pedal.
Controller	Implements the control algorithm and sends commands to the System Under Control to achieve a Goal.	The driver uses the driving wheel to turn left/right.
Control System	A system implemented to monitor and control the System Under Control enforcing desired behavior to	The driver.

	accomplish user objectives. It is composed of Controllers and Estimators.	
Environment	Anything interacting with the System Under Control that cannot be controlled directly by the Control System.	The roads, other cars, traffic, weather conditions, etc.
Estimator	Estimates State Variables from measurements produced by the System Under Control and the Environment.	The driver reading the car's speed.
Goal	Goals are specifications of desired behaviors derived from the analysis of the system requirements. They are expressions of intent for the System Under Control. A Goal has associated with it a number of Procedures which allows to split complex behavior into a sequence of simpler Goals.	The instruction "Turn right".
Goal Executor	The Goal Executor executes the Goals prepared by the Goal Scheduler.	The component of the navigator system that gives the instruction to the driver.
Goal Monitor	The Goal Monitor has two responsibilities: monitor the status of executing goals (check whether the Goal has been achieved), and coordinate a response to failure.	The component of the navigation system that verifies whether the instructions have been achieved.
Goal Planner	Responsible for producing a valid Goal Network by selecting a certain Procedure, according to system, environmental conditions, scheduling requirements, etc. The Goal Planner can either start from a single goal, or a specified (design) goal network.	The component of the navigation system that produces the sequence of instructions.
Goal Refinement	The process of breaking a high level Goal into lower level (simpler) Goals by selecting one of the Procedures assigned to the high level Goal. If there are Procedures defined, the refinement would select the appropriate Procedure given the specific circumstances.	The component of the navigation system that finds all possible intermediate instructions to drive from A to B.
Goal Scheduler	Process the Goal Network by merging the goals with overlapping time constraints.	The component of the navigation system that selects which instruction to give to the driver.
Hardware Adapter	Hardware element in the System under Control providing a measurement and command interface between the Control System and the System Under Control. It can store an history of the commands sent and of the measurements received.	The brake pedal, driving wheel, etc..
Measurement	Instantaneous evidence on the value of a state variable of the system under control obtained via hardware interfaces.	Speed from the speed gauge.

Model or State Effect Model (SEM)	Model of the Real World built on the cause-effect relations between State Variables.	Maps.
Physical State	Description of conditions ("truth data") in the system under control in terms of state variables. This may be a hardware, environment or even a software state.	True car position and speed (not the estimated position/speed).
Real-Time Goal	Goals that are created dynamically at run-time by a Controller and passed to another Controller.	Driver maintains a certain speed.
Real World	The System Under Control and the Environment.	The car, roads, traffic, weather conditions, etc.
Scenarios	Interconnected Goals that allow to achieve an objective within a given time specification.	The valid sequence(s) of instructions to drive from A to B in a temporal order.
State Variable (SV)	State Variables represent the estimated knowledge of the Real World over time.	Estimated car position and speed (System Under Control), estimated Traffic light status (Environment)
Supervisory Software	The Goal Network, Goal Planner, Scheduler, Executor, and Monitor.	The navigation system.
System Under Control (SUC)	The Hardware Adapter and Hardware (e.g. sensors and actuators) that should/can be controlled.	The car.

3. PROTOTYPE IMPLEMENTATION

3.1 Software Platform and Development Environment

The prototype has been implemented in Java using Java Native Interface (JNI) technology to access C/C++ libraries used for telescope axes simulators (based on ESO Standard Telescope Axis Controller[5]) and for pointing and tracking computations (based on SLALIB library[6]). Communication services to allow inter-process communication were provided by RabbitMQ (www.rabbitmq.com). State Machines were implemented using the Apache Commons SCXML library (commons.apache.org/scxml). Java applications were developed and executed on Microsoft Windows OS.

3.2 Model Transformations

The State Effect Models presented in the previous sections have been manually transformed into a COMODO profile[7] compliant SysML model (the stereotypes from the SysML profile are used to annotate the model for transformation)used to specify the components of the Control SW. In particular, the behavior of Controllers, Estimators, and the Goal Monitor was defined via State Machine models. The SysML model was then transformed into Java application skeleton code by the COMODO toolkit[8]. Goals, Real-time Goals, Commands, Measurements, and State Variables were also defined in the SysML model and they were translated into Java classes representing RabbitMQ messages. For these classes the serialization and deserialization methods were also generated.

3.3 Middleware Messages

In the prototype, Goals, Real-time Goals, Commands, Measurements, and State Variables are implemented as RabbitMQ messages. Each message has one or more attributes. Attributes are encoded in a buffer of bytes and specified by triplets: (name, type, value). The encoding/decoding is performed at runtime and therefore attributes can change position within a message, or new attributes can be added without the need to recompile the applications. If an attribute is deleted and the application is not recompiled, it will throw an exception.

3.4 Controllers and Estimators

Controllers and Estimators are Java applications built on top of an SCXML[9] interpreter. The SCXML interpreter executes the SCXML model generated by COMODO from the SysML State Machine. Since the SCXML model is interpreted it can also be changed at run-time without the need to recompile the application. Only the Actions and Do-Activities defined in the State Machine model have to be implemented by the developer and cannot be modified at runtime.

3.5 Hardware Adapters

The Local Control System (LCS) provides the Hardware Adapter (HWA) interface. In the prototype the main structure and dome LCSs have been implemented using simulators, one for each axis: altitude, azimuth, and the dome azimuth rotational axis. The simulators provide: the position and velocity loop, the simulation of the dynamics of the mechanical system, and the simulation of telescope azimuth axis for rotations of more than 360 degrees. The position and velocity loop controller is integrated in the form of auto-generated C-code from the Simulink model. The configuration of the Standard Telescope Axes Controller has been modified to run at 10 Hz while the reference points of the tracking trajectory are computed at 2 Hz (on the Auxiliary Telescope the reference points are computed at 20 Hz while setpoints are sent to the axes at 500 Hz).

3.6 Goal Network, Monitor, Planner and Executor

In the prototype, the Goal Network and Goal Planner have not been implemented since considered not applicable to the telescope domain and are also quite expensive to implement. They were replaced by macro-goals: a sequence of goals executed one after the other. For simplicity, Goal Scheduler and Goal Executor have been merged into the Goal Monitor application. The Goal Monitor was implemented with the same approach used for the Controllers/Estimators.

3.7 Information Flow

In the prototype Controllers, Estimators, Simulators and Goal monitor are distributed processes exchanging different types of messages: control messages like Goals, Real-time Goals and Commands, and data messages such as State Variables and Measurements. Goals are produced by the Goal Executor and received by Controllers, Estimators, and Goal Monitor. Real-Time Goals are produced by Controllers, called master Controllers, and received by other Controllers, called delegated Controllers, and the Goal Monitor. Commands are produced by the Controllers and received by the HW Adapters. State Variables are produced by Estimators and received by Controllers, Estimators, and Goal Monitor. Measurements are produced by the HW Adapters and received by the Estimators.

Figure 11 illustrates the information flow (control and data) in the architecture where Controllers B and C delegate control to Controller A. The Goal Executor sets a goal on SV A by sending a goal message to Controller A and a delegation message to both Controller B and C. Controller A starts then to send Real Time Goals to controllers B and C which in turns will issue commands to the related HW adapters. Measurements coming from the HW adapters B and C are used by the Estimators to publish estimates for SVs B and C. Finally, the Estimator of SV A produces its estimate based on SV B and C values and the Goals monitor can evaluate whether the Goal is satisfied or not.

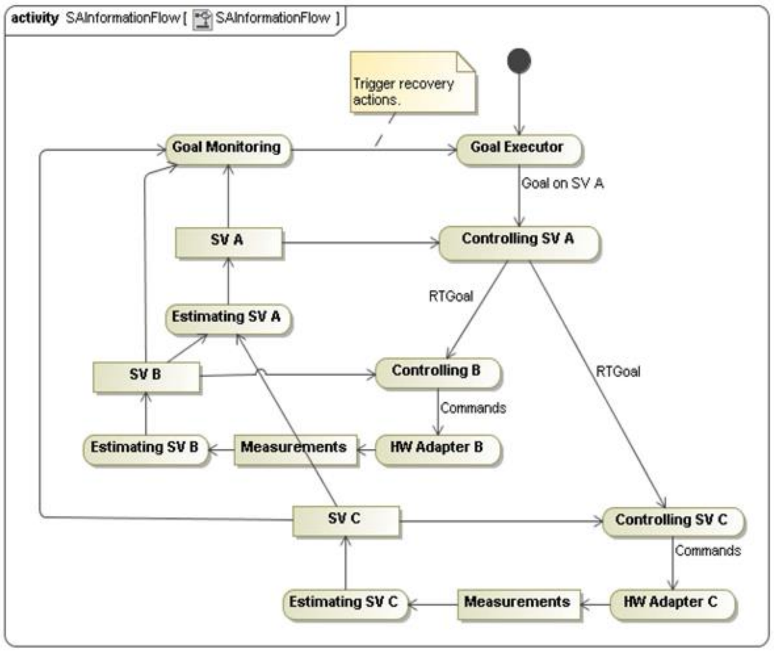
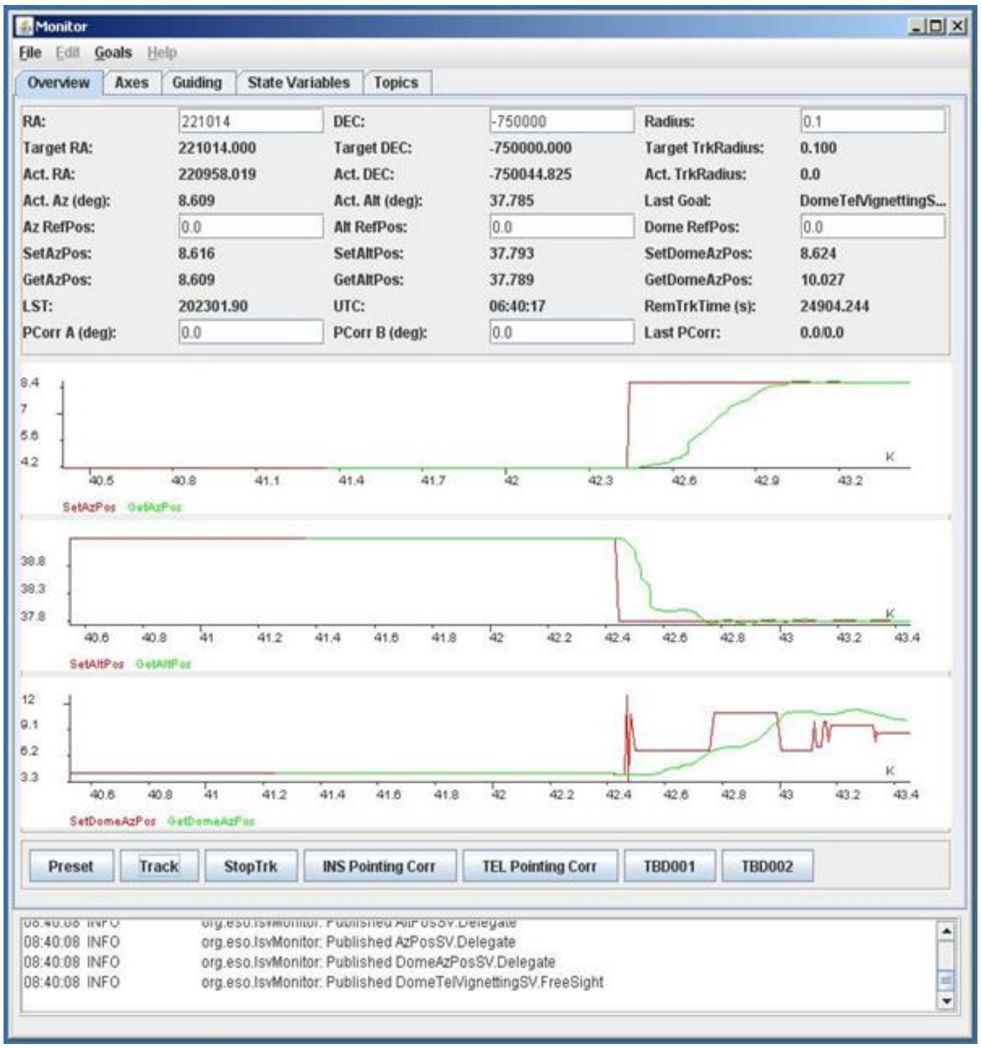


Figure 11. Information flow between Goal Monitor, Goal Executor, Controllers, Estimators and HW Adapters.

3.8 Graphical User Interface

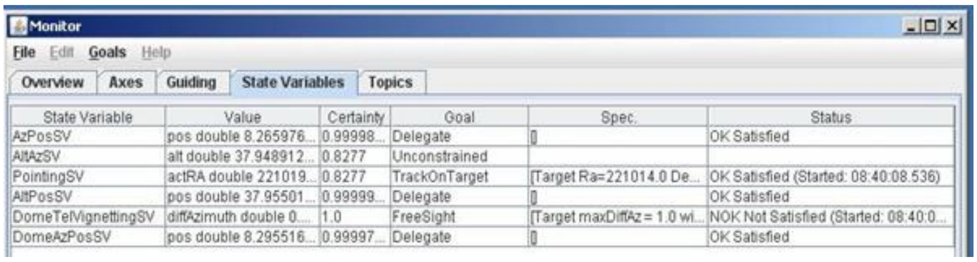
The prototype control SW can be operated using the control and monitoring GUI illustrated in Figure 12. The GUI provides one tab for different aspects of the system: Overview, Axes, Guiding, State Variables, and Topics. The overview tab can be used to monitor the overall system state (e.g. axes targets in red and actual positions in green), to enter telescope target coordinates and pointing corrections, and to start the execution of goals using the buttons. In the prototype the goal network elaboration was replaced by the concept of macro goal: a simple sequence of goals. The buttons in the GUI triggered the execution of a macro goal. For example, the Preset button triggers the execution of the AltPosSV.Delegate, AzPosSV.Delegate, and PointingSV.TransitionToTrackOnTarget goals. The Axes and Guiding tabs provide details on each simulated axis and on the guiding loop while the State Variable tab (Figure 13) shows the list of all State Variables with values, certainty, assigned goal, specification and current status (e.g. satisfied or not).


Monitor
File Edit Goals Help
Overview Axes Guiding State Variables Topics
SetAzPos GetAzPos
SetAltPos GetAltPos
SetDomeAzPos GetDomeAzPos
Preset Track StopTrk INS Pointing Corr TEL Pointing Corr TBD001 TBD002

RA:	221014	DEC:	-750000	Radius:	0.1
Target RA:	221014.000	Target DEC:	-750000.000	Target TrkRadius:	0.100
Act. RA:	220958.019	Act. DEC:	-750044.825	Act. TrkRadius:	0.0
Act. Az (deg):	8.609	Act. Alt (deg):	37.785	Last Goal:	DomeTelVignettingS...
Az RefPos:	0.0	Alt RefPos:	0.0	Dome RefPos:	0.0
SetAzPos:	8.616	SetAltPos:	37.793	SetDomeAzPos:	8.624
GetAzPos:	8.609	GetAltPos:	37.789	GetDomeAzPos:	10.027
LST:	202301.90	UTC:	06:40:17	RemTrkTime (s):	24904.244
PCorr A (deg):	0.0	PCorr B (deg):	0.0	Last PCorr:	0.0/0.0

```
08:40:08 INFO        org.eso.IsvMonitor: Published AltPosSV.Delegate
08:40:08 INFO        org.eso.IsvMonitor: Published AzPosSV.Delegate
08:40:08 INFO        org.eso.IsvMonitor: Published DomeAzPosSV.Delegate
08:40:08 INFO        org.eso.IsvMonitor: Published DomeTelVignettingSV.FreeSight
```

Figure 12. The main user interface of the prototype: The Overview tab.



State Variable	Value	Certainty	Goal	Spec.	Status
AzPosSV	pos double 8.265976...	0.99998...	Delegate	[]	OK Satisfied
AltAzSV	alt double 37.948912...	0.8277	Unconstrained		
PointingSV	actRA double 221019...	0.8277	TrackOnTarget	[Target Ra=221014.0 De...	OK Satisfied (Started: 08:40:08.536)
AltPosSV	pos double 37.95501...	0.99999...	Delegate	[]	OK Satisfied
DomeTelVignettingSV	diffAzimuth double 0....	1.0	FreeSight	[Target maxDiffAz = 1.0 wi...	NOK Not Satisfied (Started: 08:40:0...
DomeAzPosSV	pos double 8.295516...	0.99997...	Delegate	[]	OK Satisfied

Figure 13. State Variables displayed by the prototype user interface.

4. LESSONS LEARNED

This chapter lists the lessons learned when implementing and operating the prototype based on the State Analysis (SA) methodology[1].

4.1 Separation of Concerns

SA provides a clear distinction between SUC and Control System. The SUC is constituted of State Variables, State Effect Model, and HW Adapters. The Control System is composed of Goals, Estimators, Controllers. In the Very Large Telescope (VLT) SW this separation is not always clear: the state reported by the SW is a mix of the state of the control SW itself and the SUC. This mix needs to be avoided and is addressed by the goal-oriented architecture.

4.2 State Effect Model drives the Control System Design

The SEM plays a key role in understanding the SUC. It provides an end-to-end model that allows to track cause-effects relations between SVs. It is therefore important to maintain the model up-to-date and to indicate what in the model has to be implemented and what is there only for information. The model should specify whether a SV has to be estimated (i.e. Estimator is needed), whether two or more SVs can be estimated together (i.e. by the same Estimator), and whether a SV needs to be controlled (i.e. Controller is needed). It may also contain SVs which are not observable or controllable but are still in the chain of physical effects (e.g. the wave front before or after hitting a mirror). The model should not include SVs created to model entities belonging to the Control System (e.g. coordination between components) because SVs should reflect only the physical entities in the domain of the SUC. Deleted SVs should also be documented with the motivation for the removal.

4.3 Increase Scalability by combine State Variables

What drives the size of a SEM is the number of SVs and the number of effects between SVs. The number of SVs can be controlled by combining more than one SV together. SVs combination implies also the related Controller/Estimator pairs are merged into one Controller/Estimator pair. In addition, Estimators can estimate more than one SV (while a Controller can control only one SV).

In the prototype the SEM contains 10 SVs which correspond to the same number of Controllers and Estimators implemented. Each estimator and controller were deployed as an independent process.

4.4 State Variables as distributed State Functions

SVs represent the state of the SUC estimated by the Estimators. They have to be accessible at any time by the Controllers since they are used as feedback in the control loop algorithms. Depending on the control algorithm, some Controllers may also need to access the history and/or future prediction of the SV. Ideally SVs should be implemented as functions over time of the state of the SUC. These functions should be independent from the Control Software's Estimators and Controllers so that when a Controller dies the rest of the Control Software can continue to work and when the Controller is restarted it doesn't need to rebuild the history of the SVs.

In the prototype, State Variables have been implemented using pub/sub topics. This light-weight approach based only on the middleware was not ideal because it was providing only the current values of the SVs and had to deal with the late joiner problem (in a pub/sub architecture, subscribers who join late, miss messages the server already sent). A more robust approach would be to use a distributed in-memory DB, as it is available in the VLT SW.

4.5 System wide definition of Certainty

The state of the SUC can be estimated with different levels of certainty depending for example on the age of the measurements or on the ratio of working/available sensors (e.g. the number of working/available encoder heads inform about the reliability of the encoder readings). Certainty is a property of the SV and it is computed by the Estimator. SA does not provide a precise definition of the Certainty since it is considered application specific.

In the prototype, certainty was computed using the age of a measurements, the quantity of the measurements (e.g. variation from the expected frequency), additional information on the validity of the measurements (e.g. whether the axis has been initialized or not), and, in case of derived SVs, the Certainty of the source SVs.

In order to avoid confusion, there should be a unique system wide definition of Certainty and it should depend on a limited number of factors. In the telescope domain, Certainty could be defined as a function of the quality and the quantity of information available to the estimator to estimate the value of the SV. Certainty should not affect directly the SV's values. It is only an indicator of how well the state can be estimated and should be kept separate from the notion of measurement accuracy. For example, the position error between a Goal to be achieved and the estimated value is not an indication of certainty.

Estimators can use Certainty to decide to use additional sources of information to improve the estimate. Controllers can use Certainty to decide to stop achieving a Goal.

4.6 Prefer Absolute Goals to Relative Goals

Normally SA Goals are defined in terms of a given frame of reference for a given SV and they are never relative to other goals or SV values. However, there may be the need to modify the current value of SV or a parameter of an active Goal by a given offset. For example, consider the case of an instrument or a GUI sending an offset to a motor which is maintaining a given position (i.e. there is already a 'maintain position' goal on the motor position SV). This can be achieved by sending the offset request to the Goal Monitor which in turn can either modify the active goal by adding the offset to the position specified in the goal or can add the offset to the current SV value.

In the prototype, the second option has been implemented since it is more similar to what is done on the VLT where offsets are sent to modify the current status and not the intention of the original command.

As general advice, Absolute Goals should be preferred with respect to Relative Goals.

4.7 Periodic Tasks as Permanent Goals

In a Telescope Control System, there is usually the need to perform periodic monitor/control activities that last for the whole duration of an observation. These activities should not be hard-coded in the SW but specified using a Goal that has to be met for the whole observation. These types of Goals are also called Permanent Goals and they should be implemented via the usual Estimator/Controller pattern. Permanent Goals are monitored all the time. When a Permanent Goal fails, the Goal Monitor will publish the notification and the Controller or the Goal Scheduler may decide to adopt a different strategy.

4.8 Use Real-Time Goals for high performance requirements

Real-Time Goals can be used to deal with tasks that cannot be planned in advance via the Goal Network and Scheduler because of performance requirements. For example, the high-level Goal of "the Dome shall not vignet the telescope pupil" (DomeTelVignetting) needs to be refined into low-level Goals which constrain the position and velocity of the Dome and WindScreen to the position and velocity of Azimuth. The controller of the DomeTelVignetting SV sends Real-Time Goals to the Controllers of the Dome Azimuth position and Dome Windscreen position. Real-Time Goals introduce a master/worker relationship between controllers. Since the DomeTelVignetting SV is derived, its Controller can consult the Azimuth SV directly and elaborate the Goal for the Dome in real-time.

Real-Time Goals are important because they allow for creating master/worker relations between controllers and for dealing with events that, due to performance constraints, cannot be handled via the Goal Network optimization and Scheduling. Since Real-Time Goals are not checked directly by the Goal Monitor a different escalation mechanism is needed for example by informing the master controller.

4.9 Add support for Goal Merging and Multiple Goals per SV

SA foresees the possibility of merging Goals of the same type applied to the same SV at the same time (or overlapping in time). This is done during the Goal Network refinement process performed by the Goal Planner using the start/stop time information associated with each Goal. Permanent Goals and Real-Time Goals can also be merged but it must be done by the Controllers.

In the telescope domain there are scenarios where multiple Goals cannot be merged in advance. For example, the Goal of checking that the axes position is within range while the Goal tracking is active. This problem can be solved by adding the capability of applying more than one Goal to the same SV. A SV is therefore associated with a list of Goals. Before adding a new goal in the list, the following actions should be performed:

- The Goal Monitor verifies that the goal is compatible with the existing active goals (for example if the telescope is tracking it should not be possible to move an axis).
- If the goal is of the same type of a goal already in the list, the Goal Monitor replaces the existing one with the new one.
- If the goal is not of the same type of a goal already in the list, the Goal Monitor adds the goal at the end of the list.

All Goals in the list are evaluated in the order they have been inserted. Completed Goals are left in the list but not evaluated (the goal evaluation algorithm returns immediately). All Goals in the list are deleted as soon as a special Goal, called NULL Goal, is set on that SV. In order to support multiple Goals on the same SV, the Goals must allow for composition. In other words, it should be defined precisely what it means for a Controller to achieve a combination of Goals.

4.10 Centralized vs Distributed Goal Evaluation

According to SA, the Goal Monitor is the only entity able to evaluate when a Goal is satisfied or not. This approach introduces the following disadvantages:

- The Goal Monitor evaluates the goal with a delay with respect to an evaluation done directly by the Controller/Estimator.
- The algorithm to evaluate the goal is often part of (or easier to integrate with) the algorithm to control/estimate. In other words, duplication of code could be avoided (i.e. easier maintenance) if the goal evaluation is performed directly by Controllers/Estimators.
- All information needed to evaluate the goal must be available to the Goal Monitor. This includes information which could be private/local to the Controller/Estimator like configuration parameters and which are not part of the goal definition. As an alternative this information must be added in the Goal definition.
- The Goal Monitor represents a single point of failure. Especially in the case where Goal Monitor and Executor is the same entity (as it is the case in this prototype).
- The Goal Monitor alone has to evaluate all the active goals for all SVs. Instead if the evaluation is assigned to the controllers/estimators the computational load is distributed.

In the VLT case, it is the controller itself or a dedicated monitoring task running on the same computational node which decides whether the command is successful or not.

If the Goal evaluation is moved to the controllers/estimators, there is still the need of a Goal Monitor which is responsible for triggering recovery actions affecting more than one SVs.

Another important point related to Goal evaluation is the complete specification of the Goal evaluation algorithm. Consider for example the VLT ConstSpeed command to move an axis at a given speed for a given period of time. The VLT command does not specify how long it should take to reach the given speed. The VLT ConstSpeed command was translated directly to a ConstSpeed Goal which was always failing since the evaluation algorithm did not consider the transition time needed to reach the constant speed. A proper implementation of the VLT ConstSpeed command using SA Goals requires the introduction of a TransitionTo Goal to reach the required speed and a MaintainUntil goal to maintain the required speed for the given amount of time (or one Maintain Goal where a time out is given to reach the required speed).

Note, that Goals must not be arbitrarily added to a Goal Network but they must be consistent with the SEM and be refined according to the SA elaboration rules.

4.11 Goal Scheduler vs Goal Sequencer

In SA, a Goal Network is defined by a graph where nodes are the Goals and where it is specified the time when a goal should start and finish. The Goal Network is processed by a Goal Scheduler which takes care of the timing information and makes sure the Goal is started at the specified time.

In the prototype, the Goal Network and Goal Scheduler have not been implemented and the starting/finishing time for the Goals were not supported. Instead the responsibilities of the Goal Monitor were extended to include a simple sequencer able to publish a sequence of goals. When one set of goals was completed, it would publish the next set of goals. The sequence was hard coded and the only timing constraint was given by the time-outs used during the Goal evaluation. The Goal Sequencer approach introduces the following problems:

- How to handle the transition between a Goal that terminates and a new Goal that starts? In particular consider the case where a TransitionTo type of Goal is followed by a MaintainAt type of goal (for example: move a motor to 10 deg and then maintain 10 deg position for 1 min).
- How to deal with Goals which requires time synchronization (like chopping for the VLT)?

In the prototype the assumption that when a Goal is completed, the Controller maintains the last SV value until a new goal or a NULL goal (a Goal that does nothing) is received. For example, when MoveTo 10 deg goals is completed, the controller will maintain 10 deg until a new goal is received) was taken.

In addition, in order to simplify the handling of the transition between TransitionTo type of Goals and MaintainAt type of Goals, the TransitionTo type of goals could be "incorporated" in the MaintainAt type of goals simply by adding to the MaintainAt type of Goals a time-out. The time-out defines the max time to reach the desired initial SV value. Of course, the control algorithm has to be extended with the ability to reach a given initial SV value. Unfortunately, this does not solve the case of TransitionTo type of Goal followed by two or more different Goals to be processed in parallel.

Regarding absolute time constraints, instead of relying on an external scheduler, they could be handled directly by the controller (this is for example the case of chopping for the VLT). In this case the Goal has to include the specification of such constraints.

In order to avoid such problems, proper support for Goal Network and Goal Scheduler should be provided. This would allow the possibility of optimizing the Goal Network via a Goal Planner and adds a powerful way to adapt the Control System to the changing environment.

4.12 Handling of Calibrated Control

For certain telescope actuators, the initial position is determined by a calibrated look-up table which is a function of the corresponding SV. For example, the initial position of the surface of an actively controlled mirror may depend on gravity. This calibrated value would provide an initial position where a reasonably good image of a star can be taken and further analyzed. Typically, the real position of the mirror actuators is unknown and only an image provides enough evidence to determine the required correction in terms of differential forces applied to the mirror. Only when this additional evidence is available, for example from image analysis, the loop can be closed on real time data. In this scenario, the Estimator for the SV representing the mirror shape can estimate the shape using the SV representing the altitude position. Note that usually there is no need to model gravity explicitly since the gravity vector changes as a function of altitude (i.e. it is enough to model altitude with a dedicated State Variable).

If we want to have a flat shape of the mirror the “mirror shape” SV needs to be constrained by a Goal like “keep the mirror flat”. The Estimator of the “mirror shape” SV should use the altitude SV and the look-up table to compute the estimated shape. The Controller should use the “mirror shape” SV and the Goal specification to decide which forces to apply to the mirror actuators (not use directly the look-up table or altitude SV). As a consequence, the look-up table should not tell which forces should be issued depending on altitude but rather how the shape of the mirror is at a certain altitude angle.

If the shape SV is actually a derived SV (from the position of the shape actuators) then there would be another SV representing the position of the shape actuators with their corresponding Controller. The shape actuator position Controller would be delegated to the shape Controller which issues corresponding Real-Time Goals.

The mirror shape itself affects the wavefront of the target to be observed. This gives the following control choices in the design of the control system:

- The Estimator uses the wavefront SV when available to estimate better the shape of the mirror
- The shape Controller is delegated to the Controller of the wavefront SV and receives RT-Goals

In the second case the shape actuator position Controller is delegated to the controller of the wavefront or the shape Controller.

The operational mode (i.e. the Goal Network) could be different for slewing (working with calibrated values) than in observation (image analysis is available).

Calibration data is considered a separate physical State Variable which affects the measurement used to estimate the SV. A calibration SV represents a physical mode or physical characteristics; e.g. bias, where the measurement model is $m=kv + b$. The calibration SV would consist of the scale factor k and the bias b

To summarize, calibrated look-up tables should be used by Estimators and not by Controllers, different operational modes may require different delegation schema, and calibration data should be modelled with dedicated calibration SV.

4.13 Handling of Telescope Offsets

Depending on the intent, telescope offsets can be considered as pointing corrections or pointing to a different target. Pointing Corrections are enhancing the estimate of where the telescope actually points, whereas offsets are thought to point to another target. Pointing corrections are handled by injecting additional evidence into the Estimator, whereas pointing to a new target actually changes the intent and therefore the Goal. Changing the goal may induce a replacement of the Goal in the Goal Network and trigger a re-scheduling.

5. FUTURE WORK

The prototype described in the previous chapters is an example of how to use State Analysis methodology to build a goal-oriented control SW able to deal with some of the main telescope use cases such as pointing, tracking, dome vignetting, and telescope offsetting. These use cases deal with the "fast" real time part of the telescope control where the reaction time is in the order of some ms (20Hz-1Kz). In addition to these performance demanding use cases there are some tasks that have to be carried out for the preparation of the observation that requires slower reaction time.

By extending the goal-oriented approach to the observation preparation phase it would be possible to define a Goal Network that covers the full night including start-up, calibration, and shut-down or even several nights of observations. The System Under Control would become the whole Observatory. The source of information would not be limited to the instrument and telescope sensors but would include the weather forecast for the whole night, the observatory rules (e.g. rules to schedule observations, high priority targets), the maintenance constraints (e.g. mirror coating, instrument availability), etc. In this scenario the Goal Network, Planner and Scheduler would play an important role since they would adapt the astronomical observations to the forecasted conditions and constraints of the observatory. The Goal Network would not only contain Goals related to one observation but to the whole night. If, for example, a high priority target event is received, the Goal Network could be rearranged to deal with it and rescheduled. Similarly, if the weather forecast changes, some Goals for the night may need to be postponed or some Goals for the next night anticipated.

In order to achieve this level of adaptation and automatism, an approach similar to the one adopted in the autonomous automotive industry could be adopted for the specification of the requirements. In the automotive domain, requirements are organized considering three adaptation levels as illustrated in Figure 14: strategic, maneuvering, and control each with different reaction time.

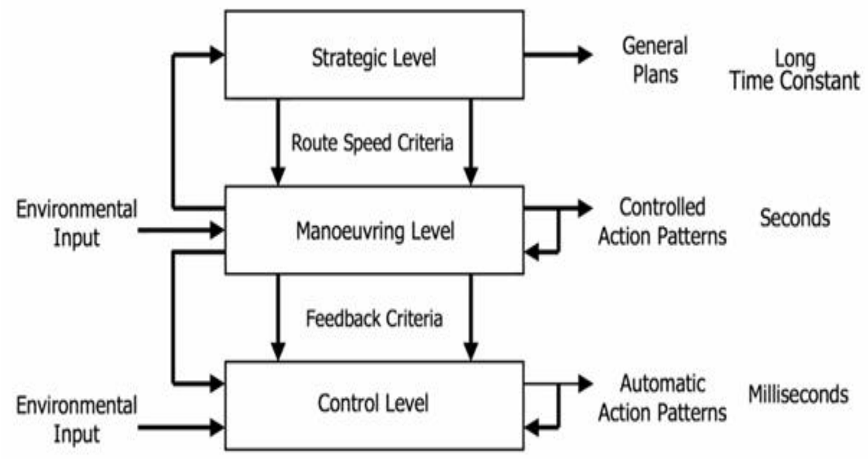


Figure 14. Adaptation levels as suggested by John Micheon[10].

In the telescope domain the three levels could be mapped to observation planning, telescope operations, and telescope control. The first level (strategic) is usually driven by the astronomers, the second (maneuvering) by the operators while the third (control) is implemented by the telescope control system.

The prototype described in this paper focuses mainly on the third level but it could be extended to level 2 and possibly level 1. This means that the scope of the System Under Control would stretch to cover the whole Observatory and that new types of Goals related to observation planning and telescope operations have to be added. The resulting Goal Network would describe one or more nights of observations and could be adapted to new Environmental Inputs via optimization algorithms implemented by the Goal Planner/Scheduler (which were not implemented in our original prototype).

A new prototype could be used to validate a reference architecture for goal-oriented telescope control systems. Such architecture, similarly to what defined for the autonomous automotive industry[11], could be composed of the functional components listed in Figure 15.

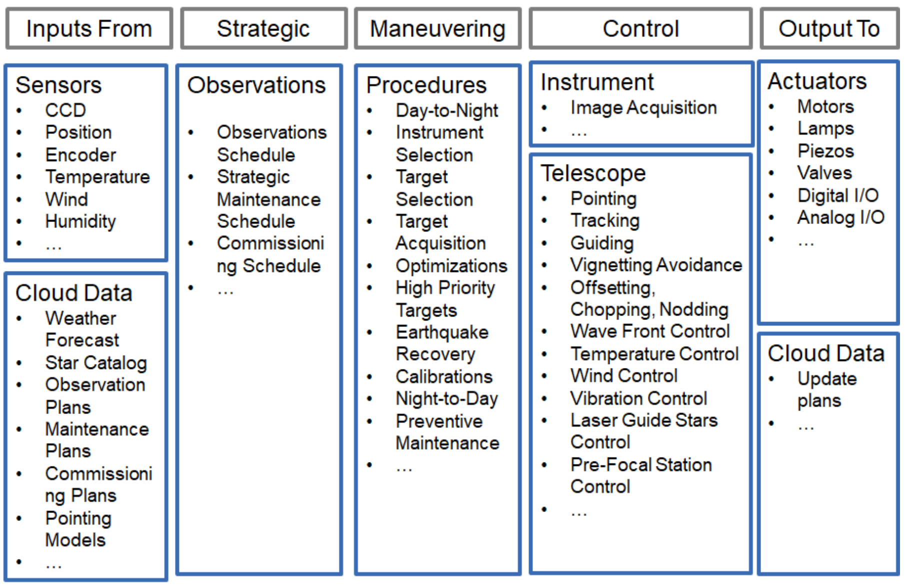


Figure 15. Possible Reference Architecture for Goal Oriented Telescope Control.

The Control aspect is dominated by RT-Goals and fast control loops processing mainly sensor measurements and driving the telescope actuators. The Strategic and Maneuvering aspects are instead handled by planning and scheduling of the Goal Network. Goal Planner uses primarily as input the information from the “Cloud Data” such as observation, maintenance, commissioning plans, and long-term weather forecast to prepare the Goal Network to be executed during the night (Strategic aspect). The Goal Network for the night is then optimized and scheduled considering Sensors and additional information coming from the “Cloud Data” (e.g. short-term weather forecast, Pointing Models, Earthquake detections, High Priority Target events, etc.). The scheduled Goals are then distributed to the State Variables to be processed by the Estimators and Controllers. Any failure or unforeseen situation that breaks the execution of Goal(s) is fed back into the “Cloud Data” and it may trigger an update of the planning and optimization of the Goal Network.

6. CONCLUSIONS

The ESO Paranal Observatory is an extremely efficient facility with a technical downtime around 2% per Unit Telescope[12]. This is due to the well optimized architecture of VLT and to the skilled telescope/instrument operators, engineers, and astronomers that know how to deal with unexpected situations in a short time and have a comprehensive maintenance strategy. In this type of context, goal-oriented architecture may not introduce relevant benefits.

The goal-oriented approach could play an important role for the new generation of telescopes, where the number of active components and interrelated control loops increases by a factor of 10 or more. In these systems, operators may get overwhelmed by the amount of information to analyze and find the best solution to unplanned conditions can become a real challenge. To help to maintain the observatory efficiency at the level of the Paranal Observatory, control software needs to be able to adapt and react autonomously to problems.

In this paper we have described the lessons learned by applying State Analysis methodology, which can be used to build goal-oriented control systems, to the implementation of a prototype for the control software of a telescope Dome and Main

Structure. In future work we would like to extend the analysis from subsystem control to observatory supervision and to define a reference architecture for goal-oriented telescope control software.

7. ACKNOWLEDGMENTS

Many thanks to Michel D. Ingham, Daniel L. Dvorak, and D. A. Wagner from the Jet Propulsion Laboratory for their help and contributions in applying State Analysis to the prototype development.

This research was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA).

REFERENCES

[1] Ingham, M., Moncada, A., Bennett, M., Rasmussen, R., "Engineering Complex Embedded Systems with State Analysis and the Mission Data System," AIAA Journal of Aerospace Computing, Information, and Communication, Vol. 2, No. 12 (2005).

[2] Choi, J. S., Coleman, A. J., Bui, B. X., Dvorak, D. L., Hutcherson, J. O., Ingham, M. D., Lee, C. Y, Wolgast, P. A., "Goal-Based Operations of an Antenna Array for Deep Space Communication," International Symposium on Artificial Intelligence, Robotics and Automation in Space (2008)

[3] Karban, R., Kornweibel, N., Dvorak, D., Ingham, M., Wagner, D., "TOWARDS A STATE BASED CONTROL ARCHITECTURE FOR LARGE TELESCOPES: LAYING A FOUNDATION AT THE VLT," Proc. of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems, (2011)

[4] Wagner, D. A., Bennet, M. B., Karban, R., Rouquette, N., Jenkins, S., Ingham, M., "An ontology for State Analysis: Formalizing the mapping to SysML," IEEE Aerospace Conference (2012)

[5] Sandrock, S., Di Lieto, N., Pettazzi, L., Erm, T., "Design and implementation of a general main axis controller for the ESO telescopes," Proc. SPIE (2012).

[6] Wallace, P. T., "The SLALIB Library," Astronomical Data Analysis Software and Systems III, A.S.P. Conference Series, Vol. 61 (1994)

[7] Chiozzi, G., Andolfato, L., Karban, R., Tejeda, A., "A UML profile for code generation of component based distributed systems," Proc. of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems, (2011)

[8] Andolfato, L., Chiozzi, G., Migliorini, N., Morales, C., "A platform independent framework for statecharts code generation," Proc. of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems (2011)

[9] SCXML W3C, https://www.w3.org/TR/scxml/

[10] Michon, J. A., "A CRITICAL VIEW OF DRIVER BEHAVIOR MODELS: WHAT DO WE KNOW, WHAT SHOULD WE DO?", in "Human behavior and traffic safety.", 485-520 (1985)

[11] Behere, S., Törngren, M., "A functional architecture for autonomous driving," Proc. First International Workshop on Automotive Software Architecture (2015)

[12] Boccas, et al, "Technical operations and maintenance activities at the Paranal Observatory," Proc. SPIE, (2018).