Goal-Oriented Architecture for Telescope Control Software

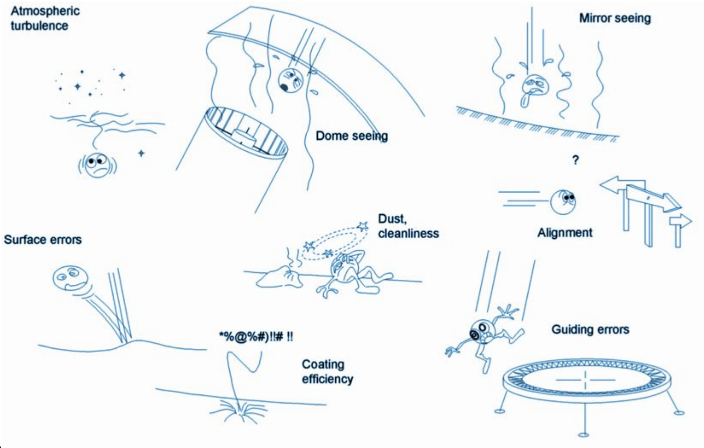
Luigi Andolfato
European Southern Observatory
Robert Karban
Jet Propulsion Laboratory, California Institute of Technology

SPIE 2020, December 2020, Virtual

Telescope Control Software State of Practice

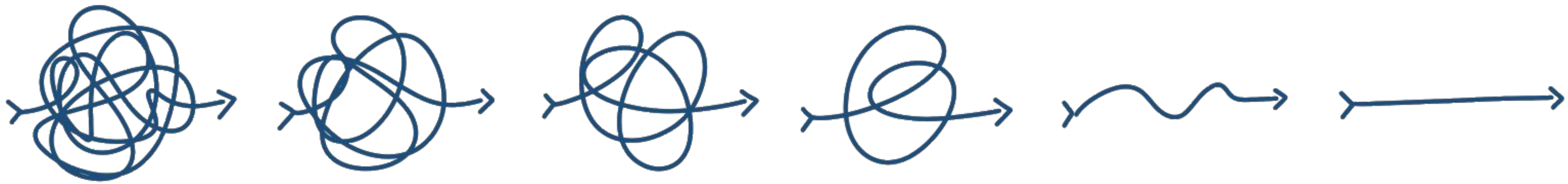
Telescope Control Software handles Very Complex Systems

- Overall **function and performance** of the telescope is allocated to the control system
- One-off experimental machines with substantial **emergent behavior**
- Many different **operational modes** and wavefront **control strategies**
- Integrated from (often contracted) subsystems which need **coordination**
- Systems **evolve** substantially over their lifetime (10-50 years) with modifications to control strategy and hardware
- **Explosion of scale for next generation of telescopes**
  I/O points: 1000s to 10000s
  Actuators/Sensors: 100s to 1000s
  Engineering data: MB/s to GB/s



Telescope Control Software faces numerous Challenges

- Increased **risk** of component **failures** due to **scale**
- Control **strategy** must be **flexible** and **adaptable** to e.g. failure or re-configuration
- **Human-in-the loop** (Operator) is getting **overwhelmed** by available information to make the right decisions
- Multitude of **interacting, distributed control loops** with hundreds of connections
- Implicit and explicit **dependencies** to consider when modifications are required
- **Difficult** to tune individual loops and **integrate them end-to-end**



Goal Oriented and State Based Architecture

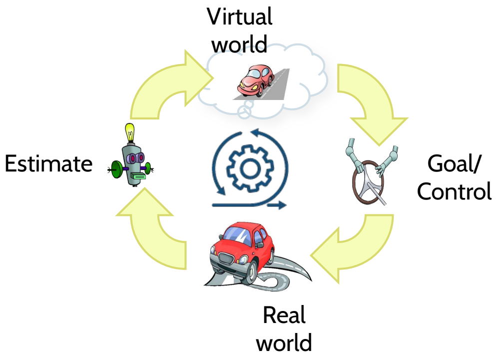
Goal-Oriented Navigation with Google Maps

- Imagine driving a car.
  As a driver you:
  - Have destinations and deadlines
  - Plan a route
  - Rely on gauges and your own senses
- By entering a destination into Google Maps:
  - Your requirement is to get from A to B
  - A route is planned based on your location and certain **constraints**
  - Based on constraints and other **information**, e.g. traffic, road-work, your location (GPS)
  - You receive directions (**Goals**) based on constraints and the state of your environment
  - Google Maps is a **Goal Planner/Executor/Monitor** and the driver a **Control System**, and the car a **System under Control**

Driver becomes a control system - with autonomy

- The driver does its best to follow those directions by
  - Monitoring the environment and the car, and interacting with the car
    Speed, location, fuel level - **State Variables**
- If you fail to achieve a goal, e.g. turn right
  - Google Maps (Goal Planner) will **reschedule** and give you a new goal
- If you replace yourself with a robot
  - Nothing really changes for goal oriented operation

“Say **what** to do **not how** to do it”

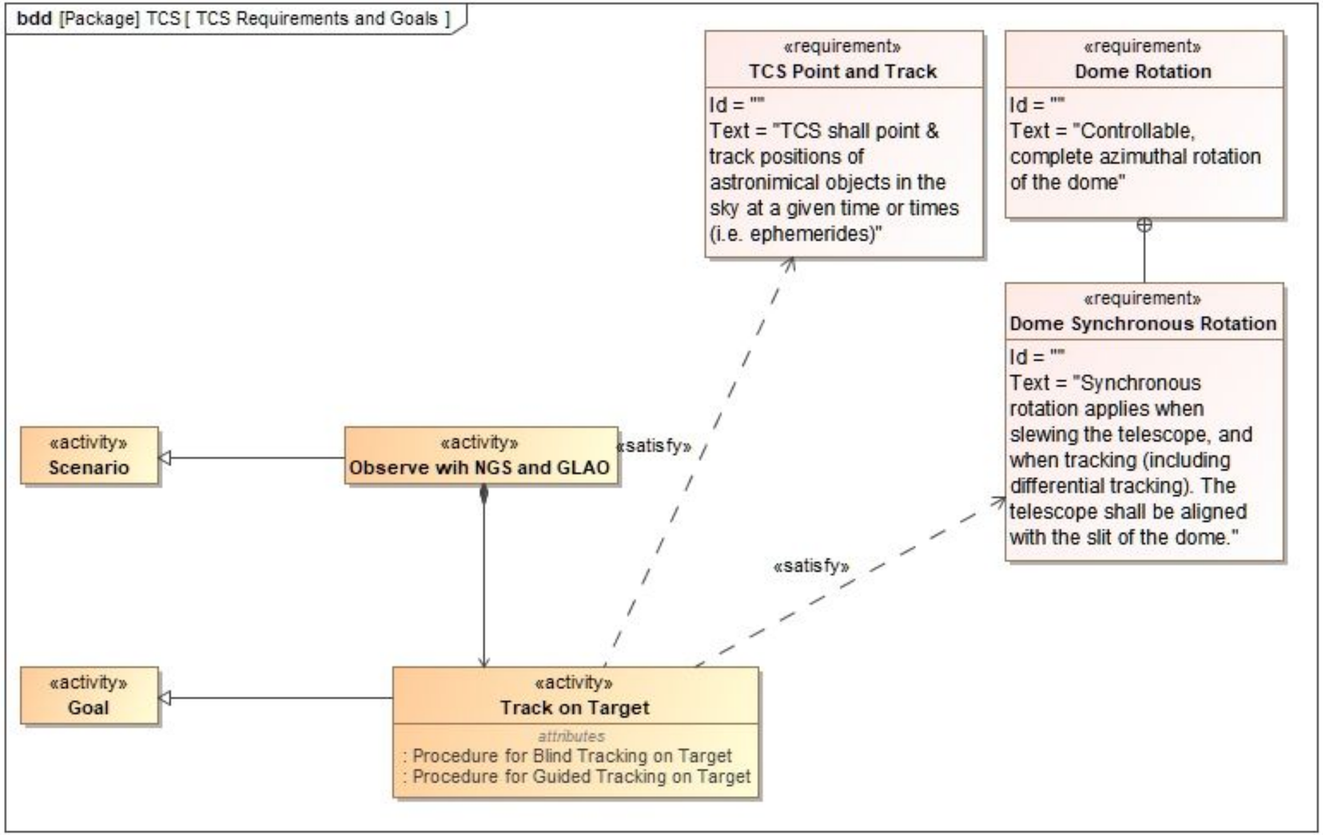


Objectives of a Goal-Oriented Control Architecture

- Capture an executable set of requirements expressed as Goals
- Provide guiding principles for control software design to implement control strategies 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

Framework for Goal Oriented System Modeling and Analysis

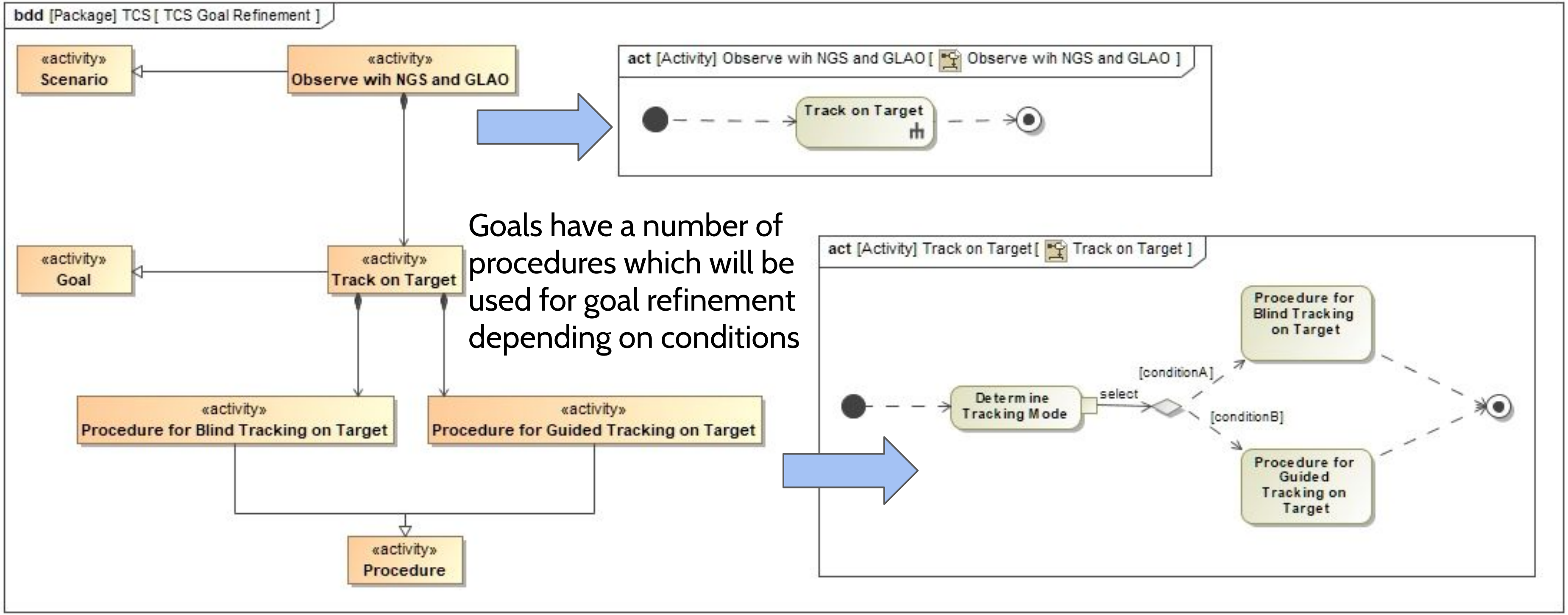
Goals and Scenarios satisfy Requirements



Goals are specifications of desired behaviors derived from analysis of the requirements.

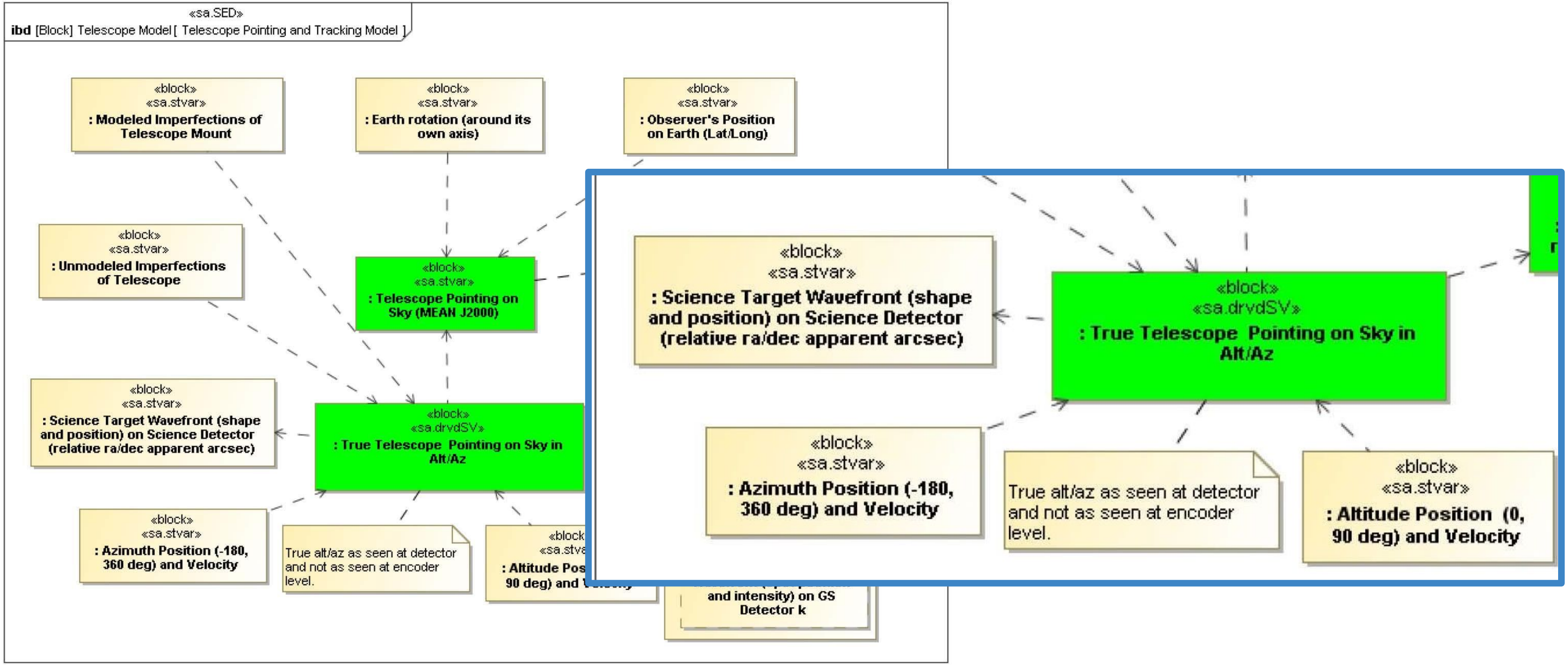
A **Scenario** is an exhaustive description of intended system behavior specified as a set of interconnected Goals and times.

Goals are refined by Procedures

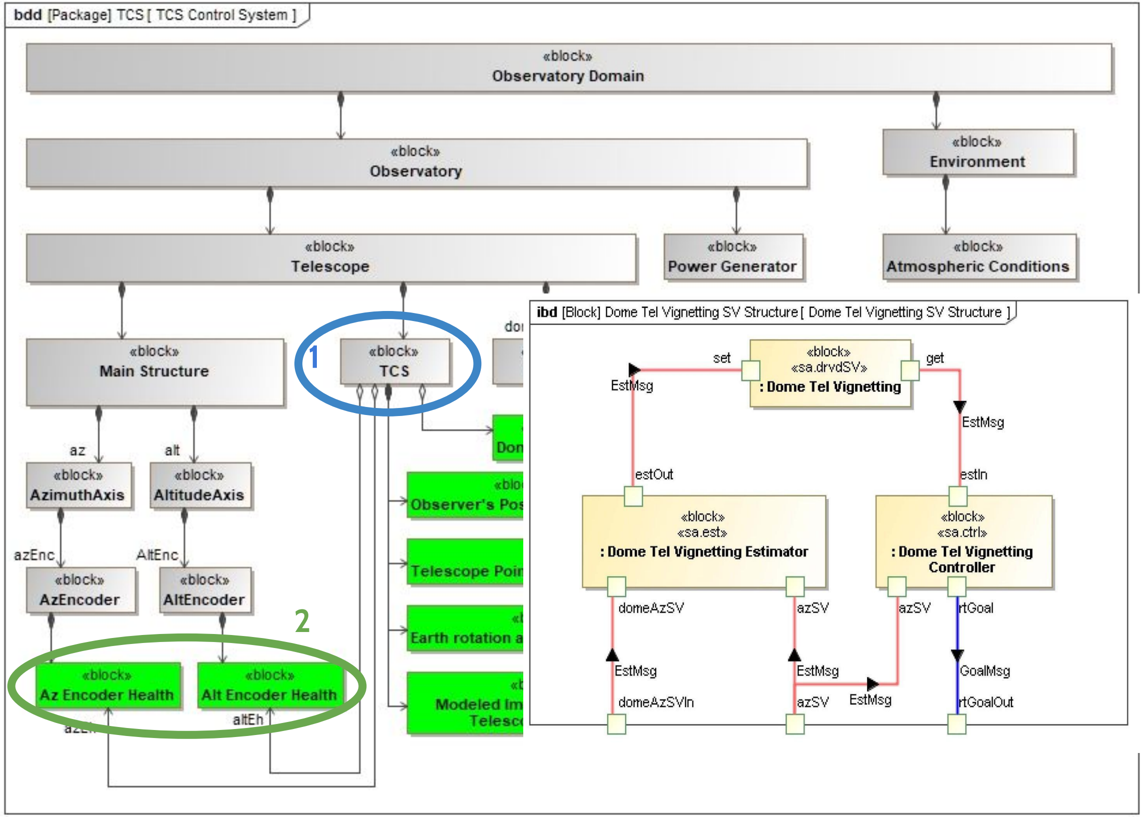


Goals have a number of procedures which will be used for goal refinement depending on conditions

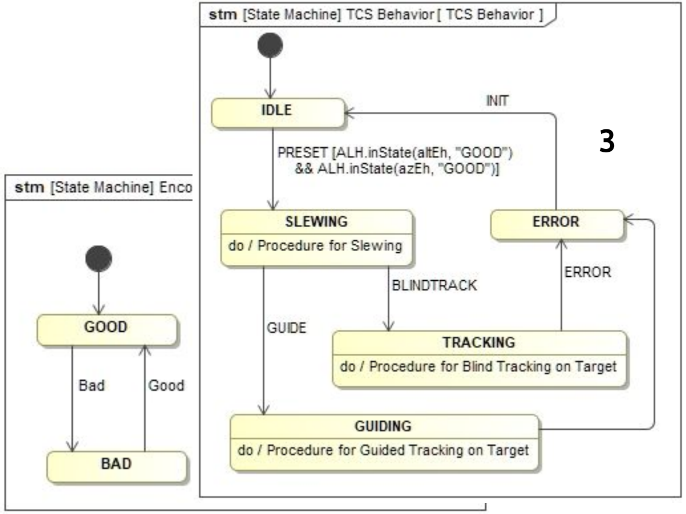
Identify relevant State Variables based on Effects



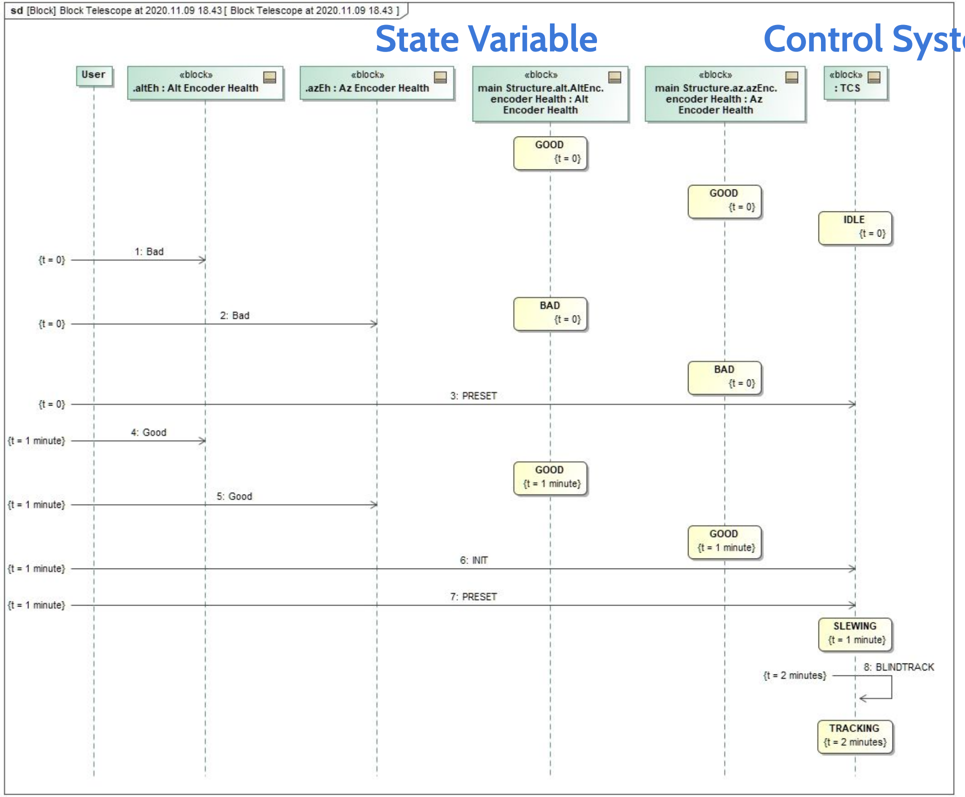
Goal-Oriented Control System Architecture



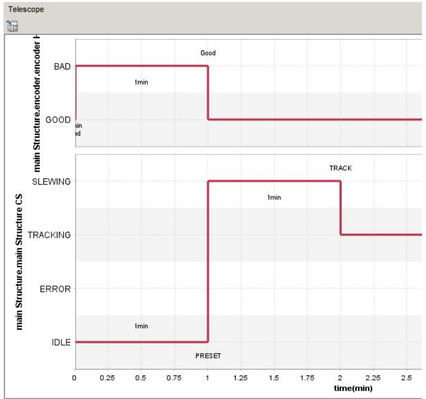
- Architecture breakdown with control systems referring to **State Variables (2)**
- **Control Systems (1)** are composed of Controllers, Estimators, HW Adapters
- Specify **component behavior** (3) and models of State Variables



System Level Simulation Validates Requirements



- Simulate the scenarios
- Express requirements more precisely
- Check if requirements make sense
- Check consistency and logic
- Perform simple analysis concerning timing and dynamic resource budgets
- Understand roles of different components.



Prototype Design and Implementation

Requirements and Objectives

- Proof of concept for **goal-oriented design and implementation** for the control of **dome and main structure** - guided by State Analysis (SA) architectural principles
- **Focus on function** (not performance) to demonstrate the **interaction of components** and validate **goal-oriented operational** approach
- Demonstrate **suitability to analyze** and model domain concepts of **ground based astronomy**: pointing, tracking, offsetting, and dome control.

Design and Implementation Platform

Application Design

- State Machine for Life-Cycle Management of Controllers and Estimators
  - Based on SCXML engine interpreter
- Do Activities for
  - Domain specific goal handling
  - Interaction with hardware (commands and measurements)

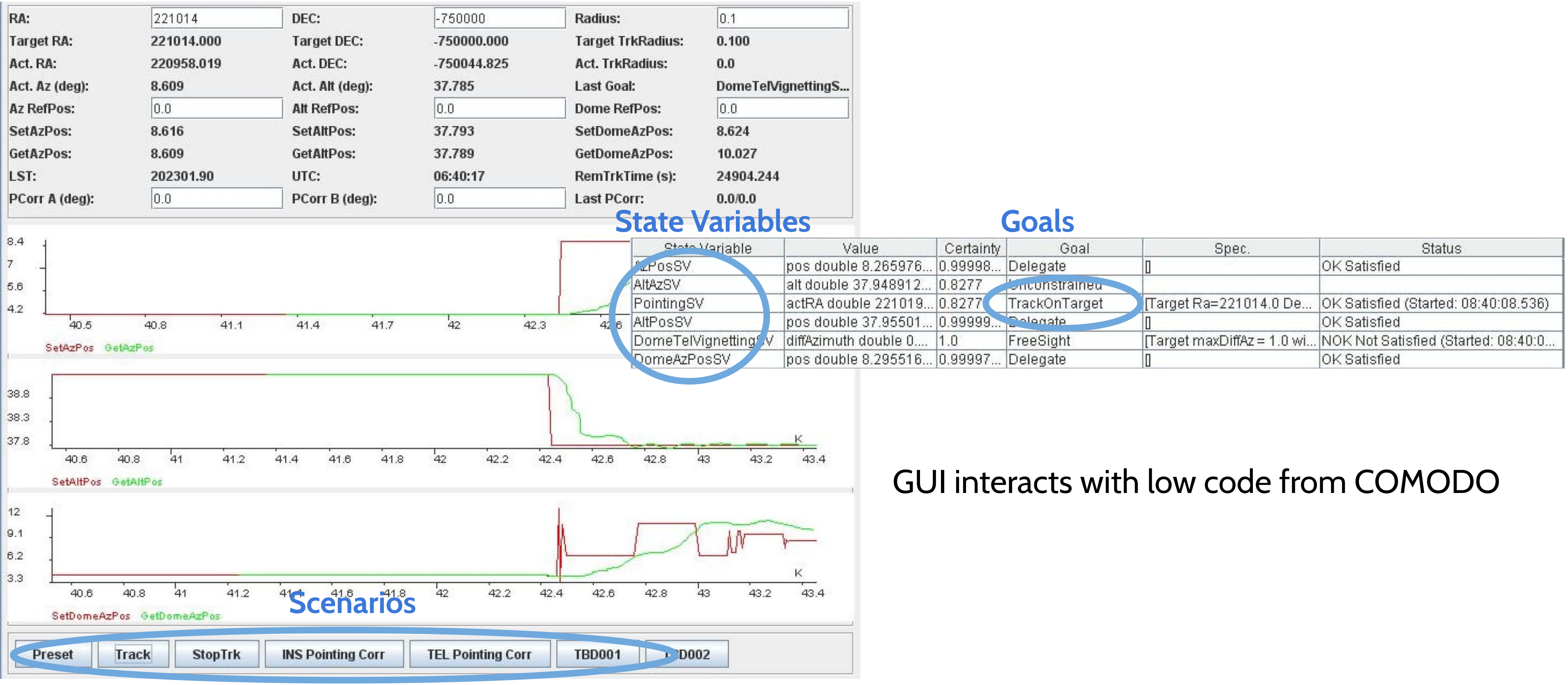
Software Platform and Development Environment

- Programming language: Java and C/C++ on Windows
- Messaging: RabbitMQ
- Slalib for astronomical calculations and standard ESO axis controller (generated from Simulink)

Model Transformations

- Conceptual Design to Low Level Design
- Low Level design to Low Code using COMODO

GUIs for Goal-Oriented Operations



GUI interacts with low code from COMODO

Conclusions

End-to-End model

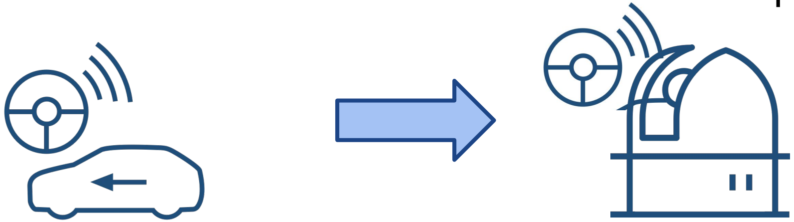
- State effect models represents the end-to-end knowledge
- Enable reconfiguration/rescheduling of goals to adjust to a new situation (dynamically at run-time)
- Enables End-to-End tuning
- Consistent System, Software Models, and Code

Separation of architectural concerns of control system and system under control

- SVs pertain to System Under Control only
- Goals pertain to control only
- Goal-oriented architecture guides the design allowing for a better integration of operational needs with the low level control. This facilitate the adaptation/reaction to a changing environment.
- State-Based patterns (Control Diamond) allow to extend the control system architecture consistently and systematically

Future Work

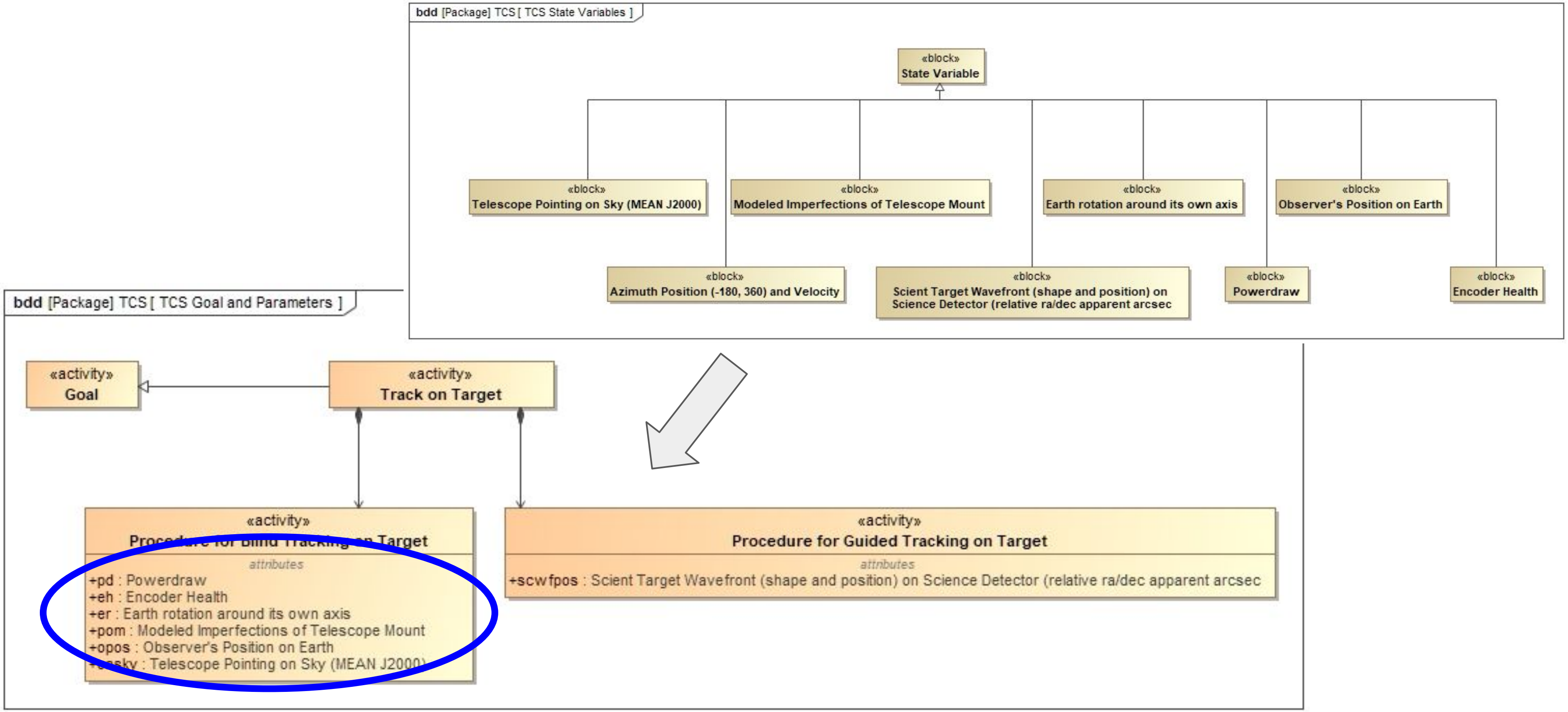
- Capture the whole Astronomical Environment in the System Model
  - Physics; e.g. pointing tracking
  - Rules; e.g. do not point to the sun
  - Environment; e.g. weather conditions
- Specify Architectural Layers more explicitly
  - Observation planning
  - Instrument/Telescope Operation
  - Motion Control
- Establish a Reference Framework for Goal-Oriented Telescope Architecture



Questions?

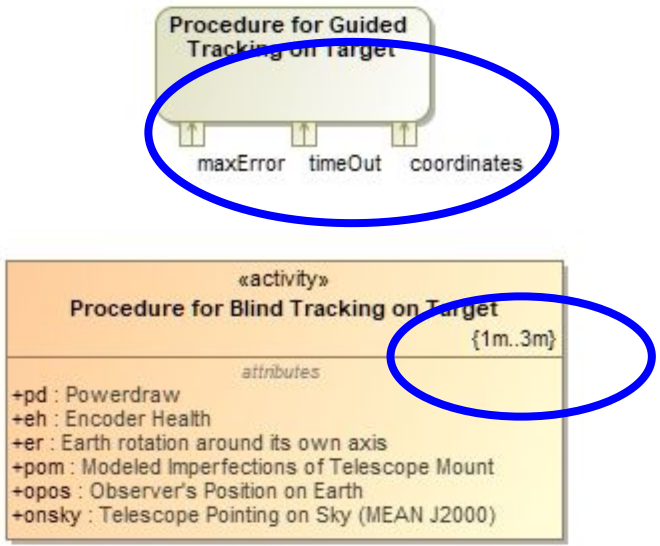
Backup

Relate State Variables to Goals and Procedures

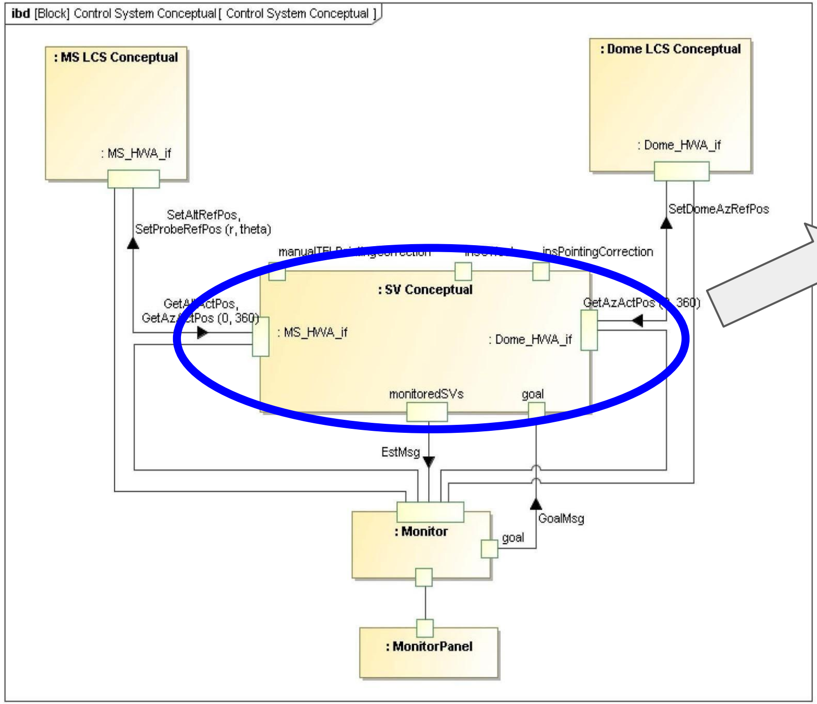


Specify Procedures and Goals

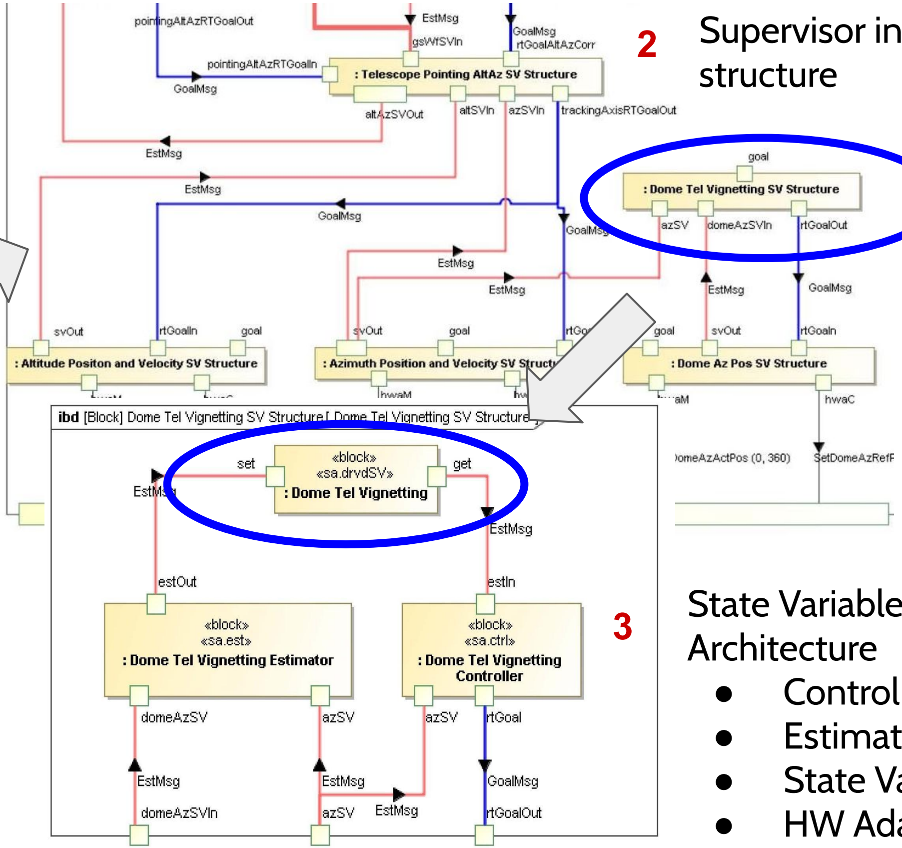
- Add duration constraints to express time constraints



Conceptual Control System Architecture



1 Supervisor and Local Control Systems



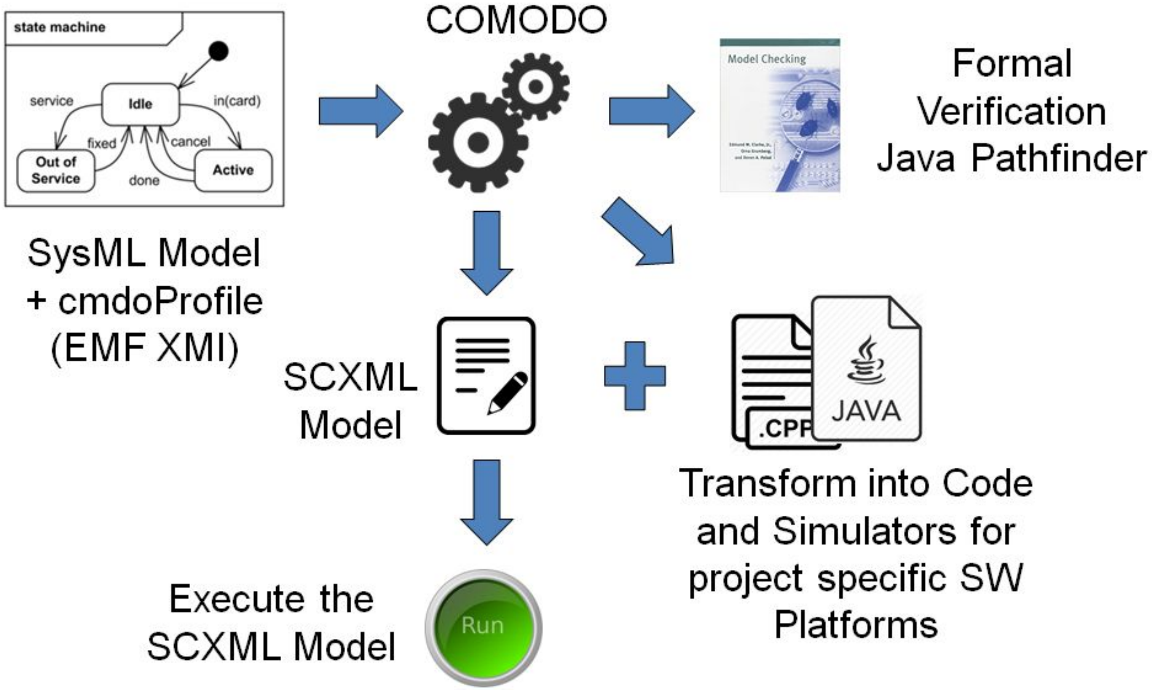
2 Supervisor internal structure

3 State Variable Micro Architecture

- Controller
- Estimator
- State Variable
- HW Adapter

Application Code Generation with COMODO

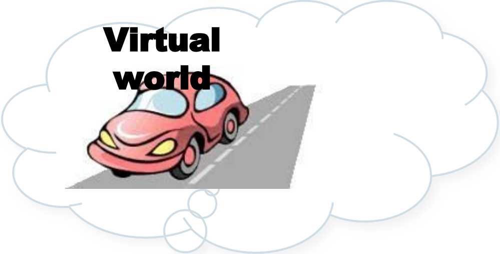
COMODO provides model-to-text transformation from state machine models to Java/C++ applications based on Apache Commons SCXML engine.



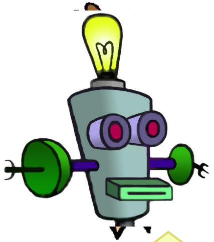
Goal Oriented “Control Diamond”

Our model of how the world works helps us make sense of our senses

Virtual world



We react, not to things as they are, but rather to things as we perceive them



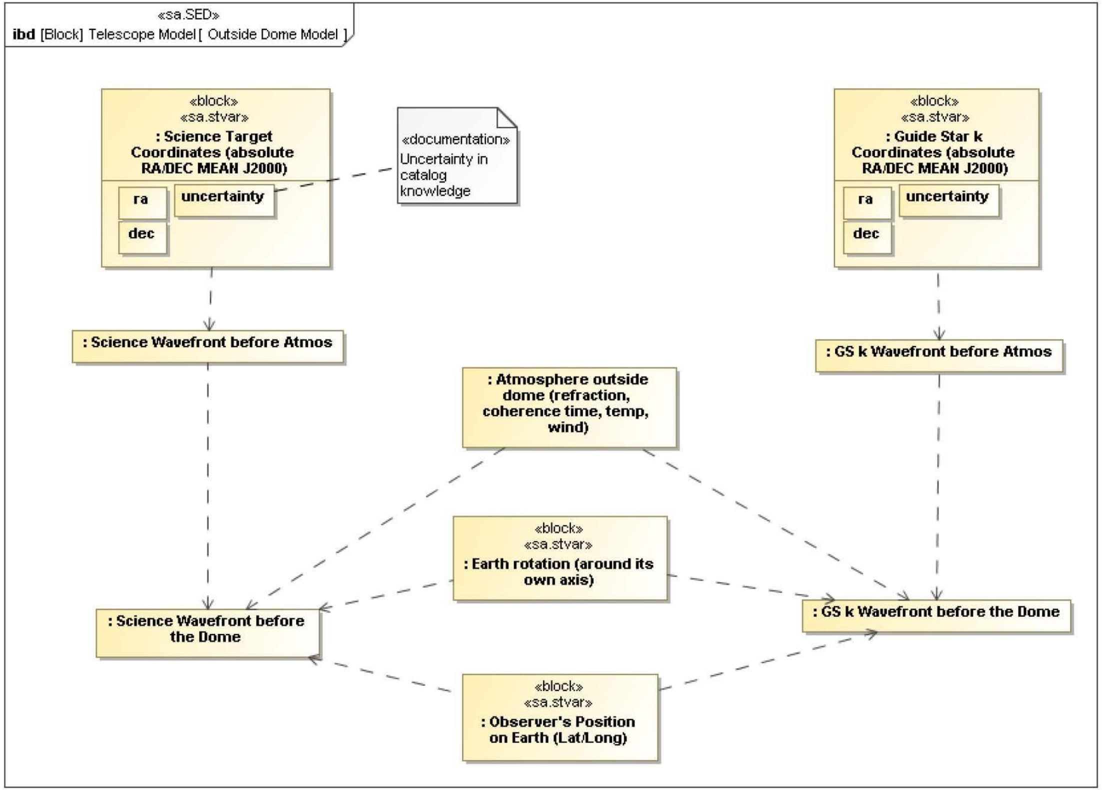
Similar behaviors

Real world

We see, not what is, but what we perceive — with a little help from our senses

Our actions are guided by what we expect them to do, given what we know

State Effects Model - Dome



TCS conceptual architecture and Control Diamond pattern for Dome Vignetting control