Jet Propulsion Laboratory
California Institute of Technology

# A Model-Based System Engineering Approach to Modern System Architectures

**Robert Karban**
*Jet Propulsion Laboratory, California Institute of Technology*

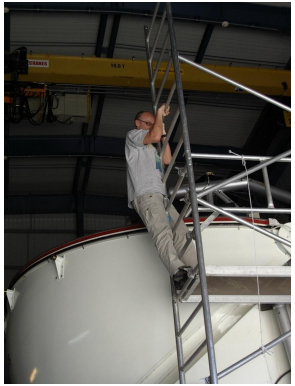
**SKA, March 2021, Virtual**

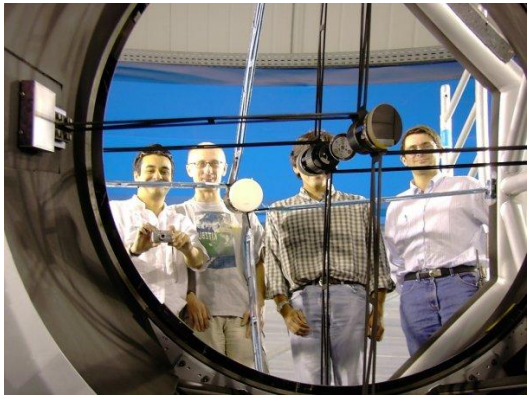
*Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.*
*The views and opinions of contributors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.*

© 2020 California Institute of Technology. Government sponsorship acknowledged

# Who is Robert?

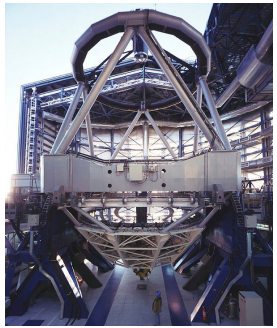
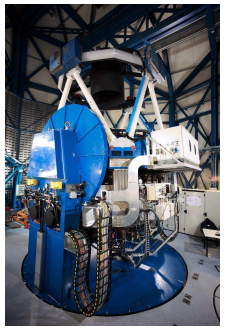
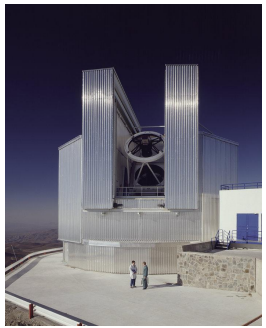
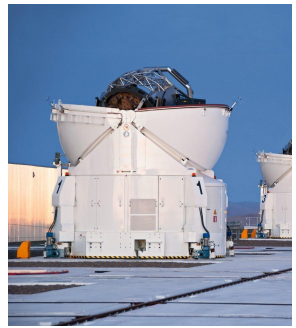
- CAE Systems and Software Environment Chief Engineer at JPL
- Member of INCOSE, SysML RTF, SysMLv2
- Formerly Control System/Software Engineer and Architect at:
  - European Southern Observatory (ESO) – Germany, Chile
  - CERN – Switzerland/France
  - Siemens Healthcare - Austria
- M.Sc. Computer Science (Austria)



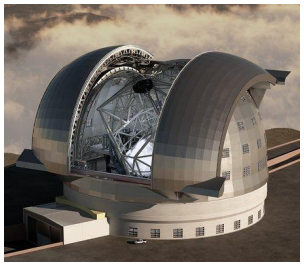
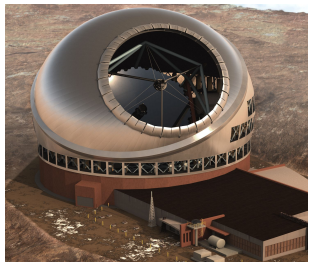



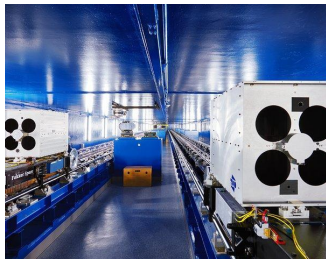

# Telescopes and Instruments Robert worked on

- 4 x Very Large Telescope – 8m
- 4 x Auxiliary Telescope – 1.8m
- 1 x Very Large Interferometer – 130m - 200m
- 2 x Survey Telescope – 4m
- 55 x Radio Telescope – 12m - 15m
- 1 x 3.6m Telescope – 3.6m
- PRIMA and APE instruments
- 1 x NTT – 4m
- 1 x ELT – 39m
- 1 x TMT – 30m

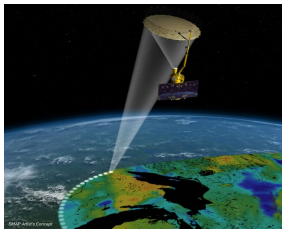


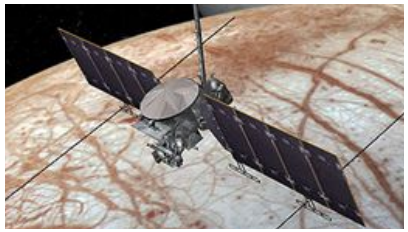




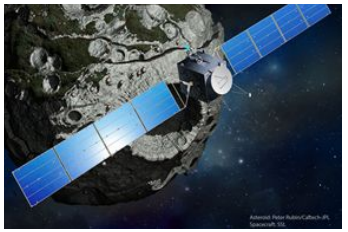


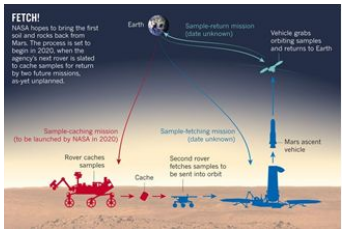
# Space Missions Robert contributed to

- Europa Clipper
- Europa Lander Mission Concept
- Mars 2020
- Mars Sample Return
- Psyche
- SMAP









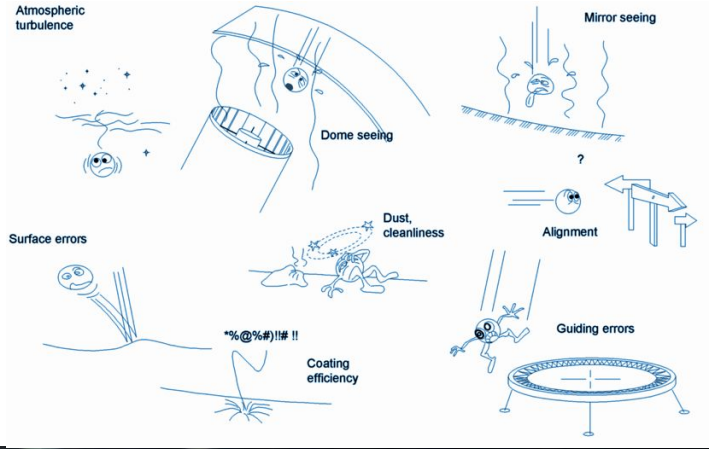

Pre-Decisional Information – For Planning and Discussion Purposes Only

# Goal-Oriented Architecture

## Telescope Control Software State of Practice

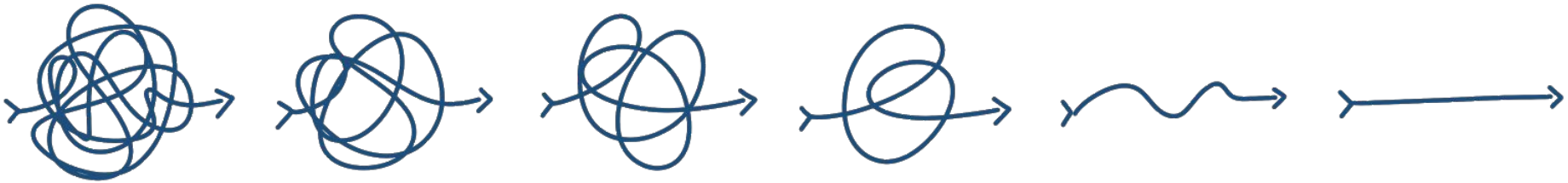
# Telescope Control Software handles Very Complex Systems

- 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 Architecture

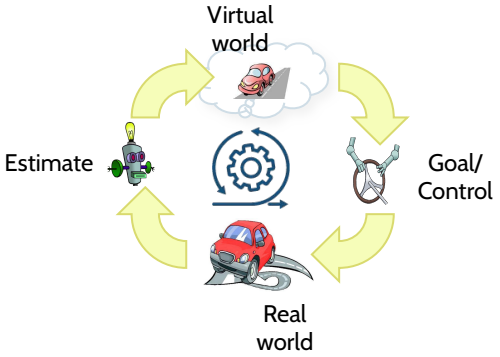
## Goals and States

# 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

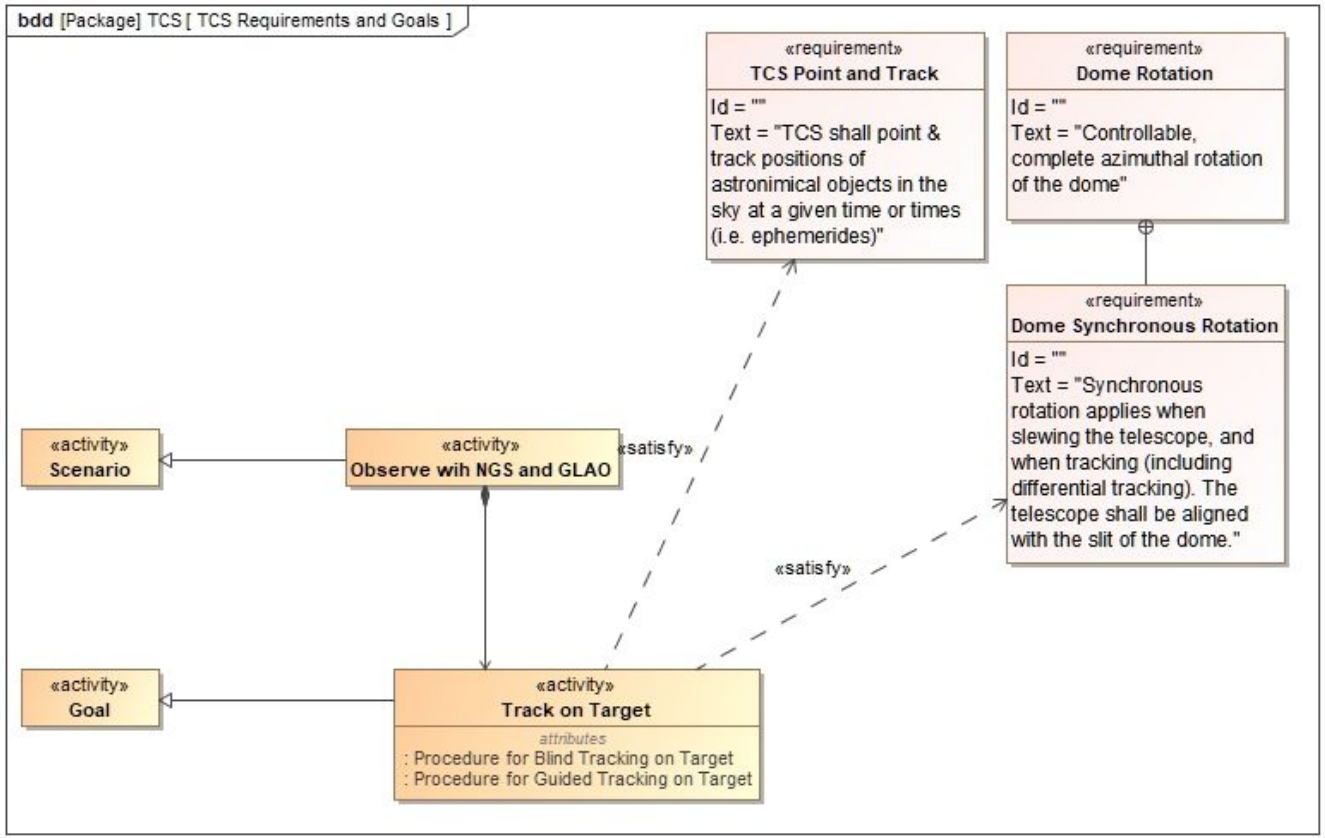



"Say **what** to do **not how** to do it"

# Goal-Oriented Architecture

## Framework for System Modeling and Analysis

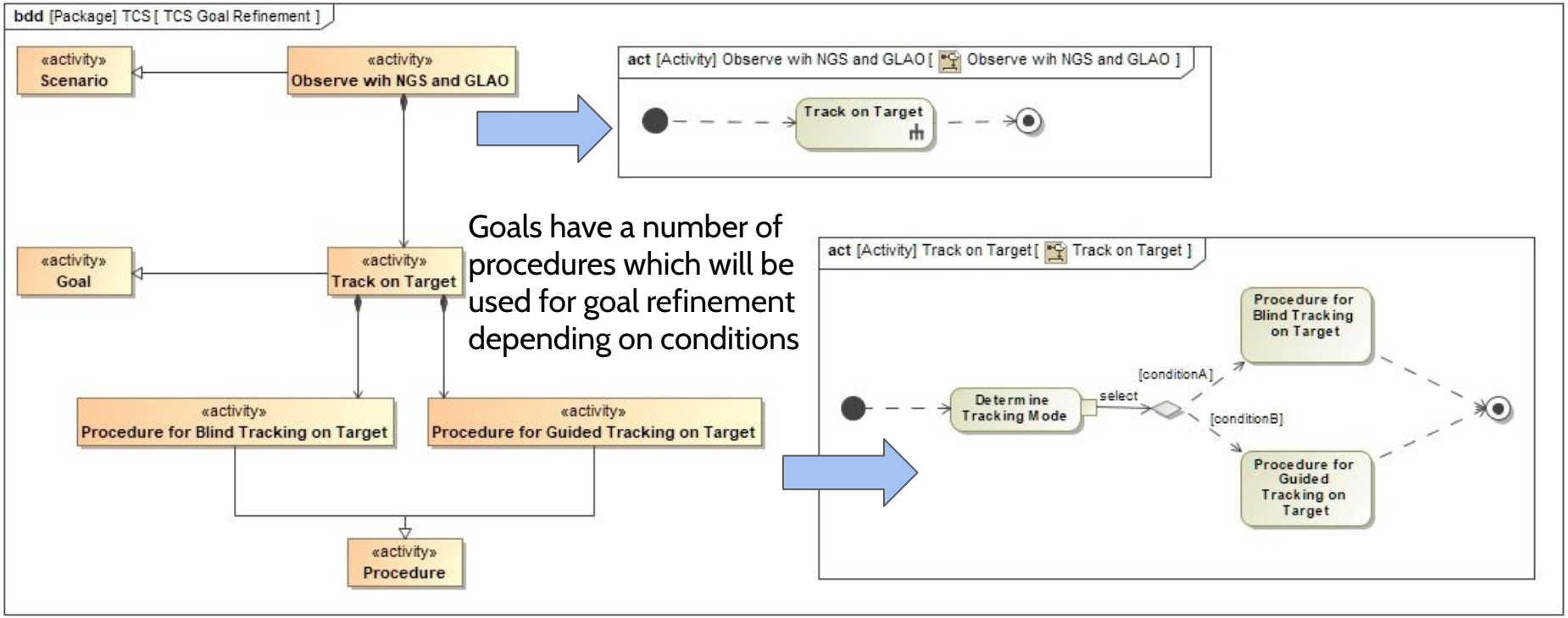
# Goals and Scenarios satisfy Requirements




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

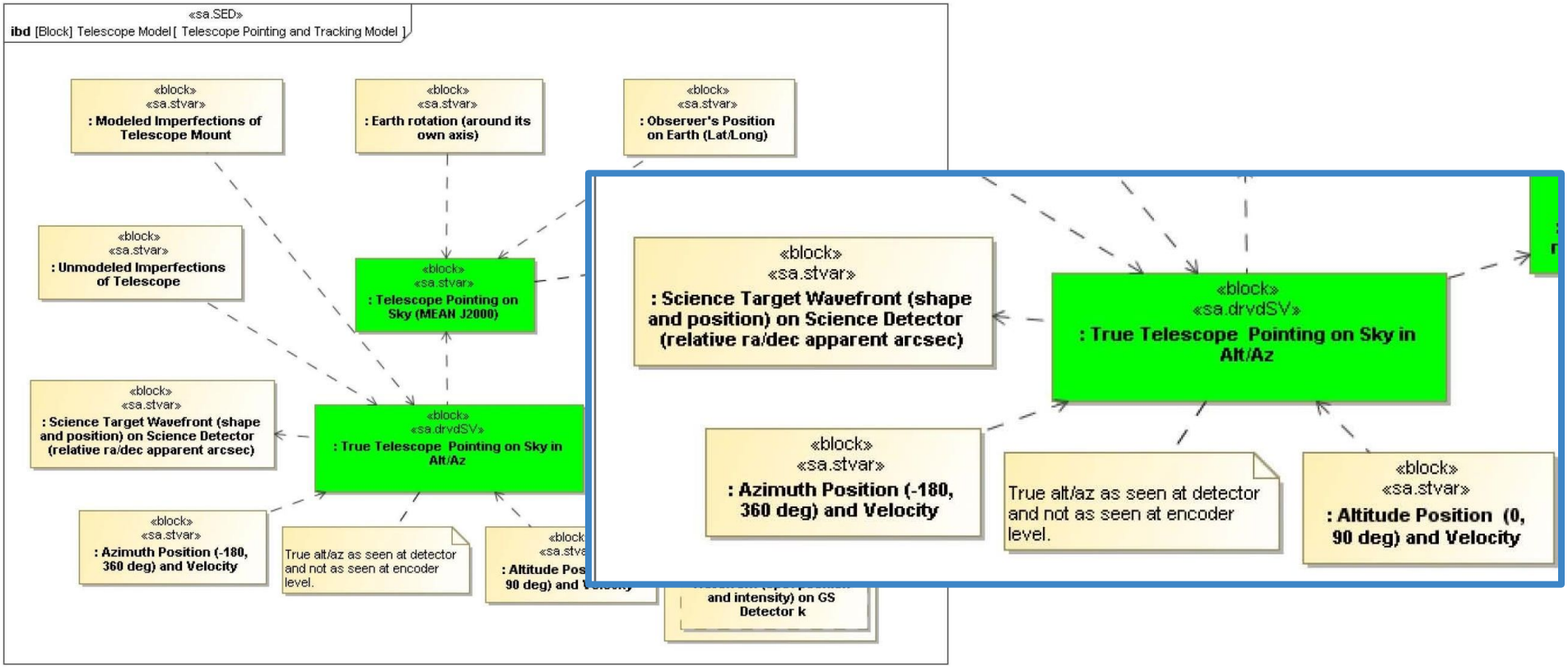
A **Scenario** is an exhaustive description of intended system behavior specified as a set of interconnected Goals and times.

# Goals are refined by Procedures

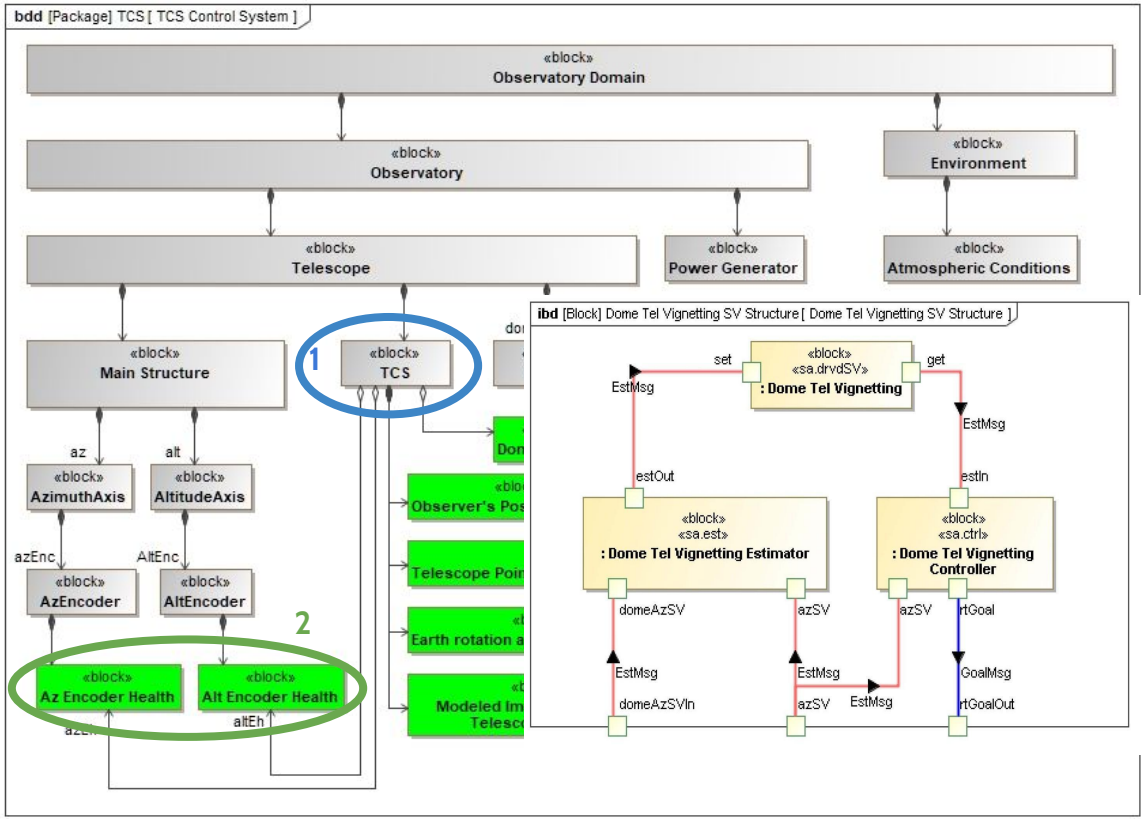



Goals have a number of procedures which will be used for goal refinement depending on conditions

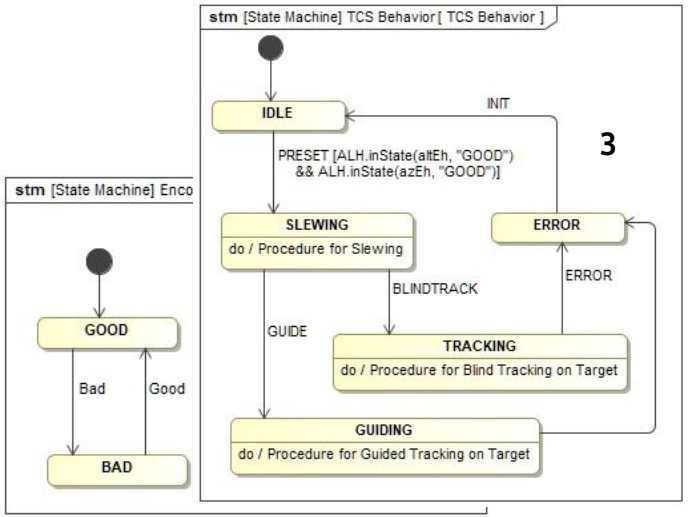
# Identify relevant State Variables based on Effects



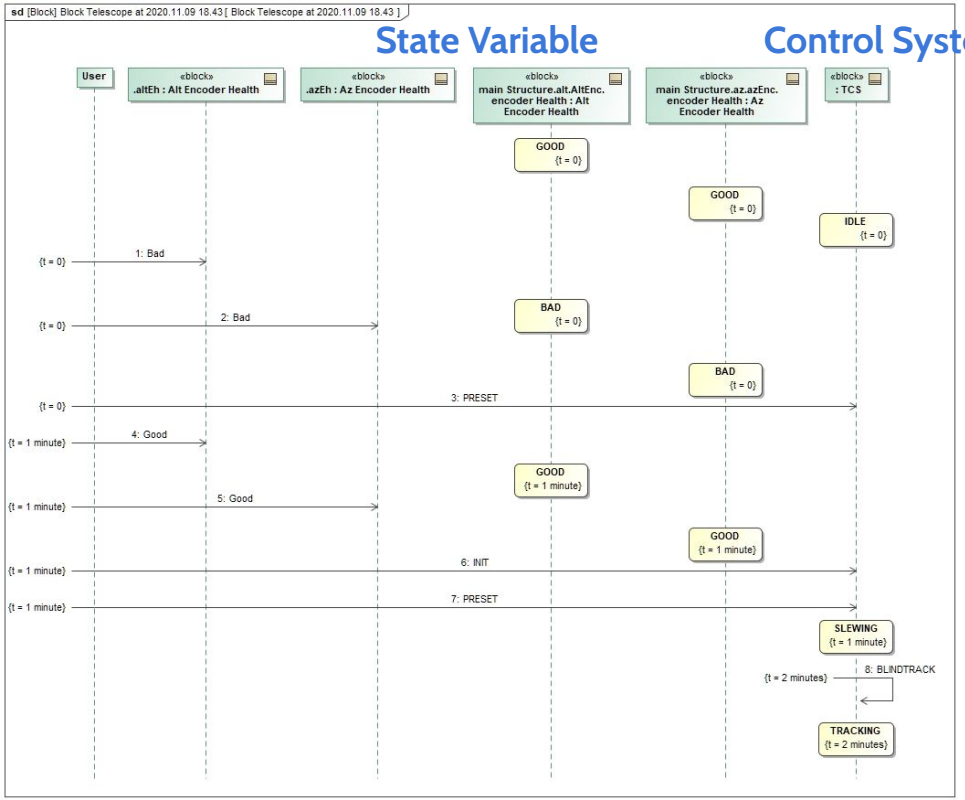
# Goal-Oriented Control System Architecture



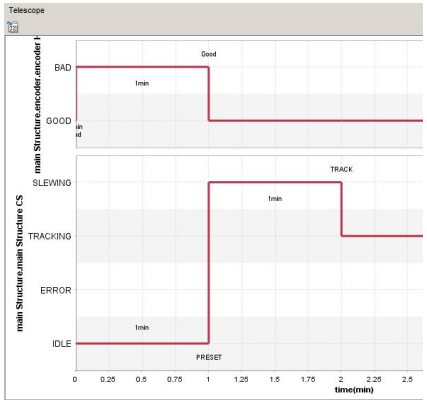

- Architecture breakdown with control systems referring to **State Variables** (2)
- **Control Systems** (1) are composed of Controllers, Estimators, HW Adapters
- Specify **component behavior** (3) and models of State Variables



# System Level Simulation Validates Requirements



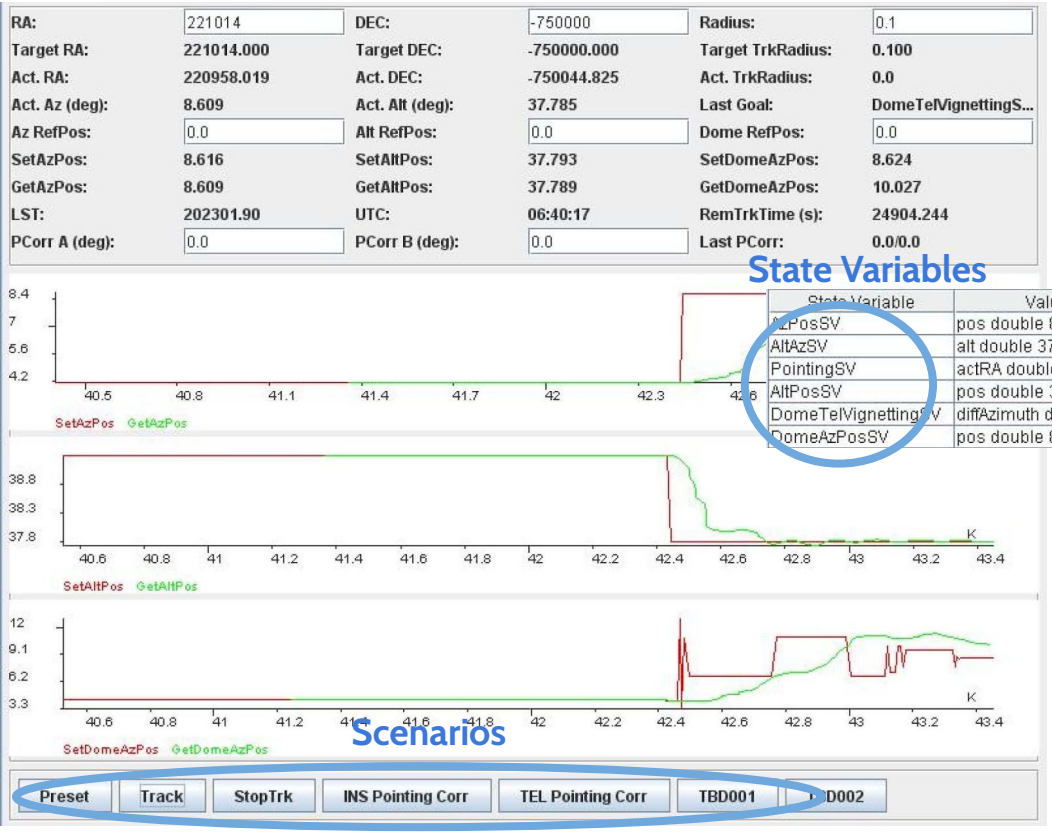

- Simulate the scenarios
- Express requirements more precisely
- Check if requirements make sense
- Check consistency and logic
- Perform simple analysis concerning timing and dynamic resource budgets
- Understand roles of different components.



# Goal-Oriented Architecture

## Prototype Design and Implementation

# GUIs for Goal-Oriented Operations




State Variables

Goals

| 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 |

Scenarios

GUI interacts with low code from COMODO

# Key Takeaways

**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

# Executable System Models

## The practitioner perspective

# Why an executable Model (at System Level)?

**D*escribe* a domain or system under study or to *specify* a (business, software and/or hardware) system to be built.**

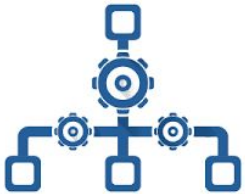
- Gain system **understanding** before or while building the system
- Specify parameters and behavior based on requirements and **validate** them in a System Model
- Express requirements more **precisely**
- **Analyze** how the as-designed system reacts to user interaction, predefined testing data and execution scenarios, timing and dynamic resource budgets

# Why an executable Model (at System Level)?

- **Explore** desirable and undesirable behaviors of a system
- **Check** consistency and logic of design
- **Understand** roles of the different components
- Enable (semantic) **consistency** across different **transformed** artifacts such as code, documentation, simulation, and analysis

# Objectives of an End-to-End Goal Based Executability Framework

- Produce a system **model** of the control architecture enabling **autonomy**
- Capture an **executable set of requirements**
- Clarify the understanding of **control architecture and operations**
- Perform **analysis** of operational scenarios and component behavior
- Enable **transformation**



# Executable System Models

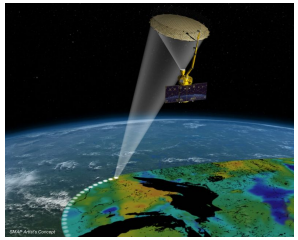
## Current State of Practice

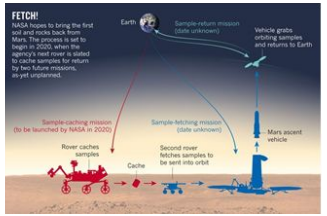
# Selected JPL Applications

- Capture
  - Operational Scenarios
  - Fault Protection
  - Flight Software Behavior
- Model
  - Activities, State Machines, Parametrics, Sequence diagrams
- Produce
  - Interface Control Documents
  - Functional Design Documents
  - Analysis reports (e.g. Timelines)
  - Code

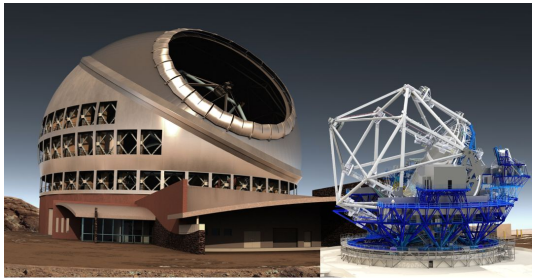
**Thirty Meter Telescope**, SMAP, Psyche,

Europa Lander Mission Concept, Mars Sample Return





# TMT applies “Hybrid” Systems Engineering Approach



## Traditional SE

- Clear, defined deliverables
- Easily accessible
- Shallow learning curve
- Simple traceability

## MBSE

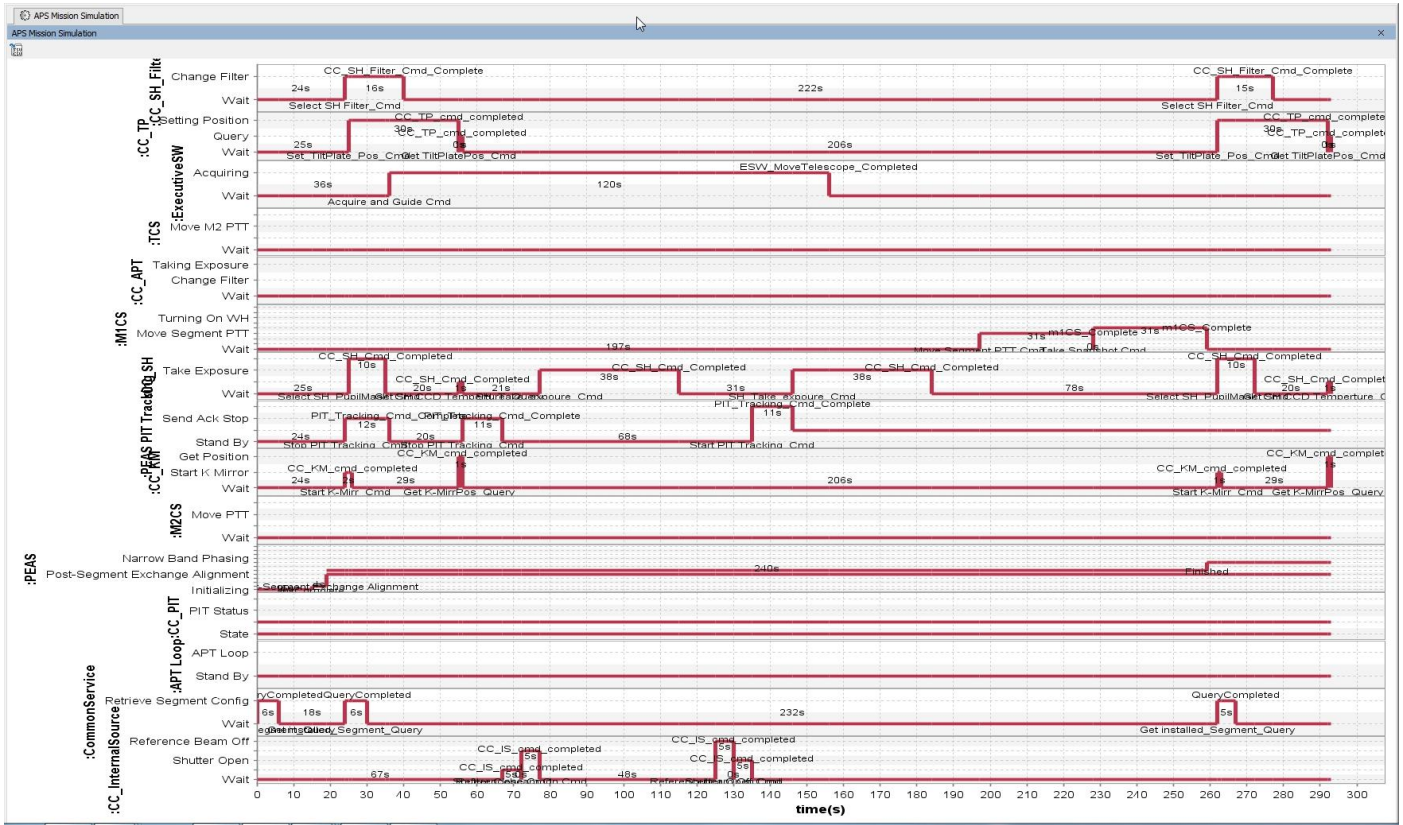
- Understanding behaviors of a system
- “Rich” capability to represent complex systems

**Exploit the advantages of each approach**

# TMT MBSE Objectives

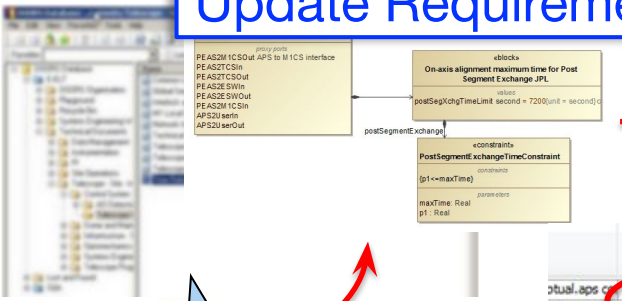
- **Capture** operational behavior, Identify involved subsystems, Identify interfaces and interactions among subsystems
- **Develop/refine** timing requirements for algorithms and external interface commands
- Create an **executable SysML model** with state machines and activities to specify behavior
- Use the model to **analyze the system design and verify requirements** on power consumption, mass, duration, pointing errors, ...
- Produce **engineering documents**
- Use **standard languages and techniques, and COTS tools where practical** to avoid custom software development

# Sample TMT timeline



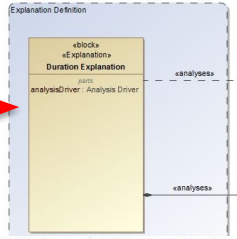
# TMT Analysis workflow

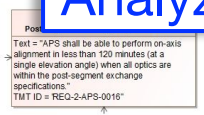
## Update Requirements 2





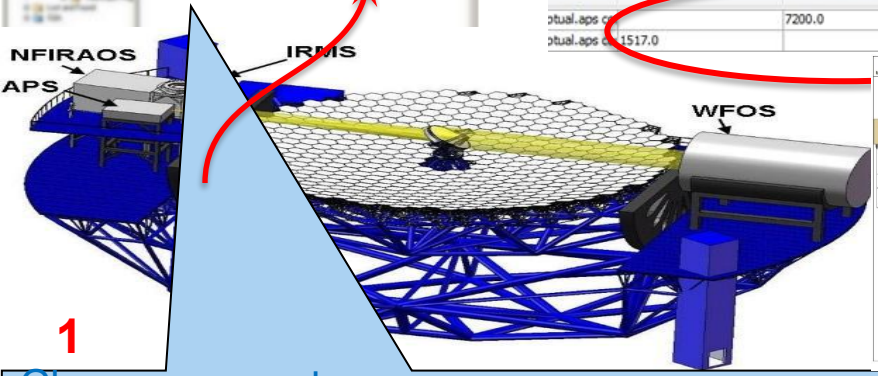

## Analyze Conceptual Design 3



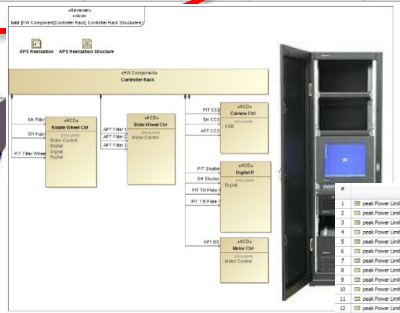


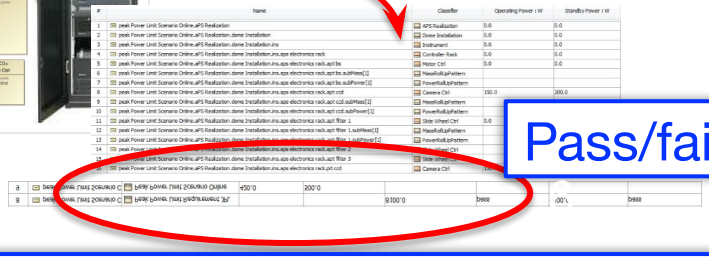
| | T Final : Real | Post Seg Xchg Time Limit : | Post Segment Exchange : Post Segment Exchange Time Constraint | Off Axis Measurement Steps : Integer | Off Axis Map Points : Integer | RB Dit : Integer | Phasing Dit : Integer |
|---|---|---|---|---|---|---|---|
| tual.aps c | | 7200.0 | pass | | | | |
| tual.aps c | 1517.0 | | | 6 | 7 | 45 | 20 |




1

## Analyze Realization Design/Specification 4





## Pass/fail

## Change request

Max duration Post-segment exchange: ~~7200s~~ 5000s
Number of exposures of 45s 4 ~~6~~
Max peak power consumption in dome: 8.5kw ~~8.1kw~~
Number of motors with 50W 10 13

## Produce Engineering Documents 5

# Systems Model is developed according to ESEM using Cookbook Patterns

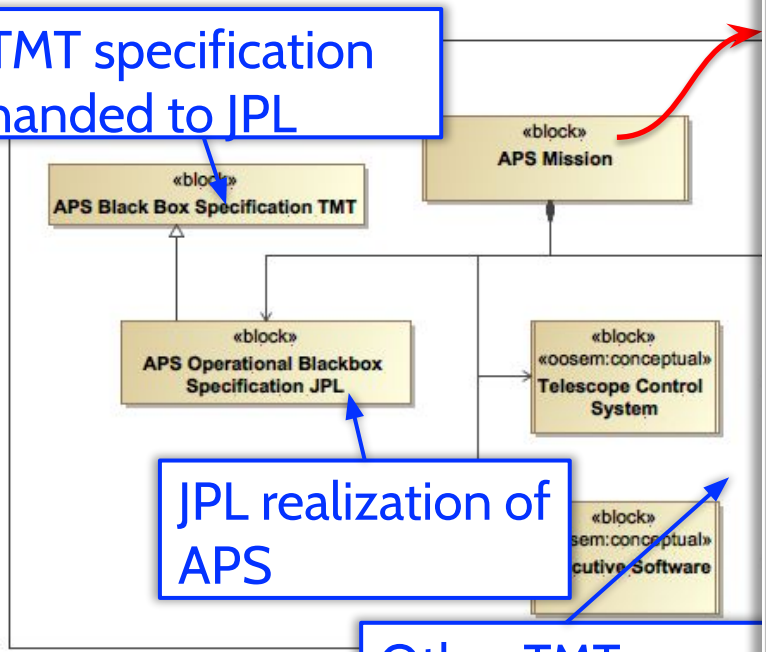
- Define APS Mission boundaries
- Elaborate Conceptual Architecture
- Capture Component Behavior and Characteristics
- Specify Interactions between Components
- Run Analyses

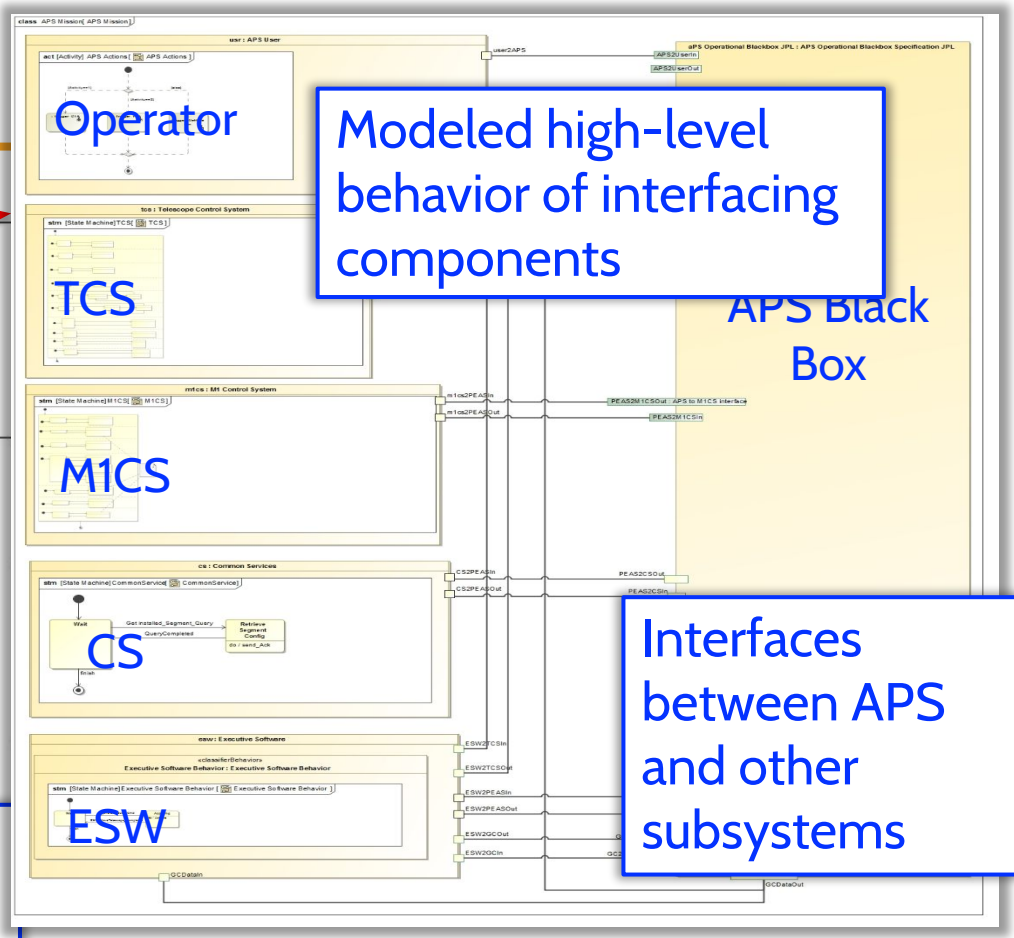
# APS Mission

TMT specification handed to JPL

JPL realization of APS

Other TMT Subsystems



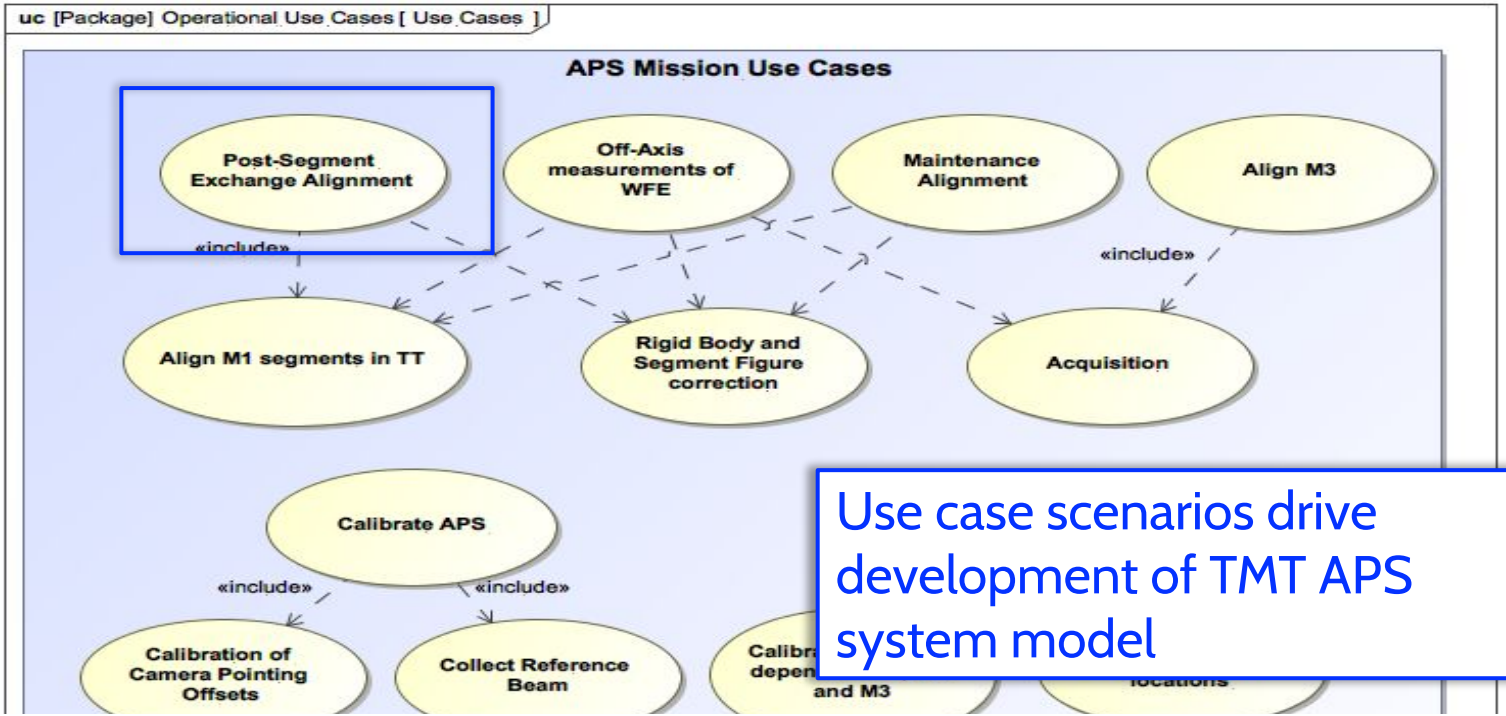




Modeled high-level behavior of interfacing components

APS Black Box

Interfaces between APS and other subsystems

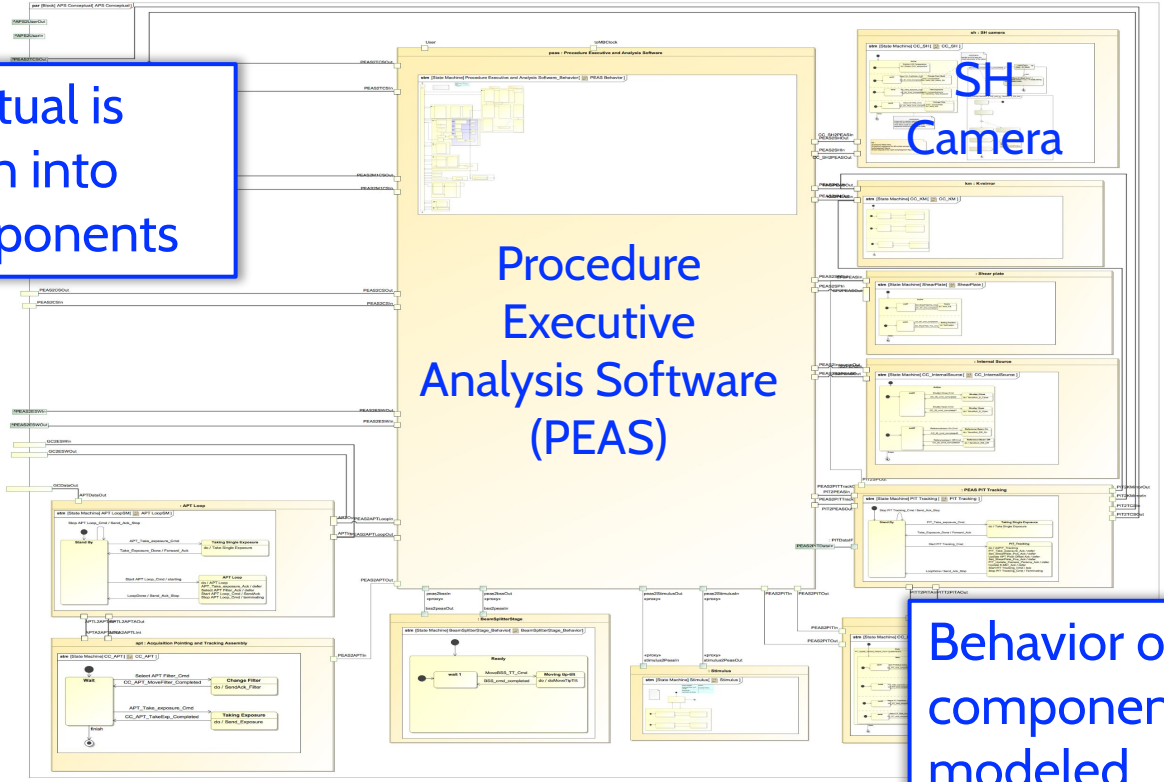
# Use Cases




Use case scenarios drive development of TMT APS system model

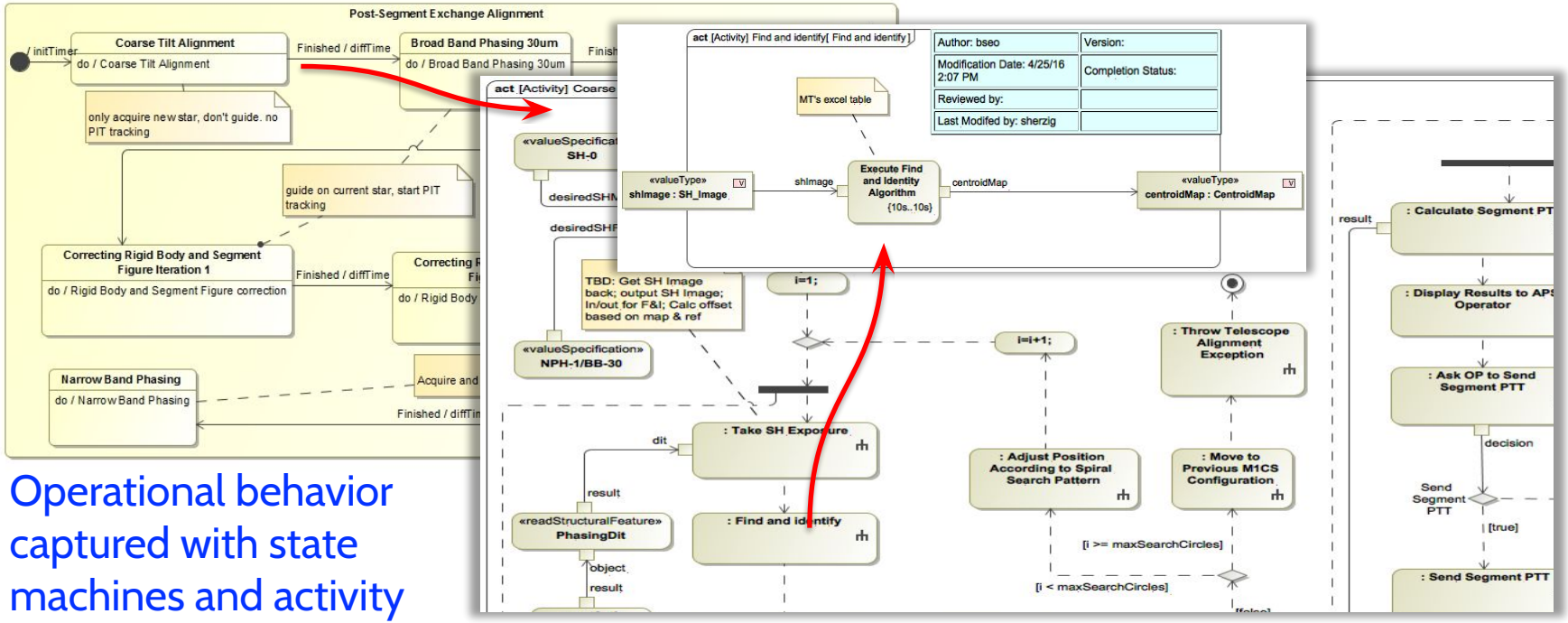
# Conceptual Architecture

APS conceptual is broken down into several components



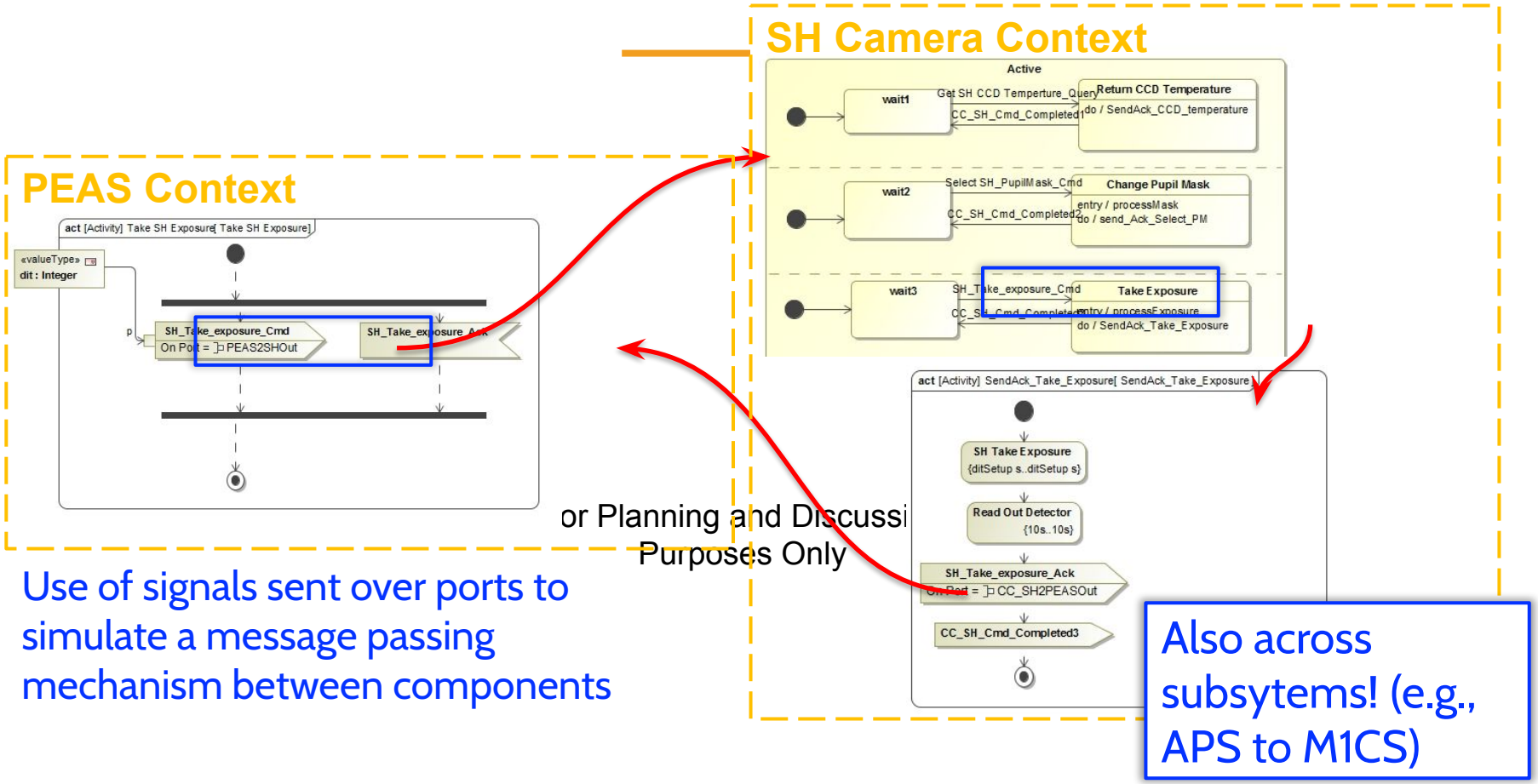

Behavior of all components modeled

# Capture Component Behavior and Characteristics




Operational behavior captured with state machines and activity models

# Interactions Between Components



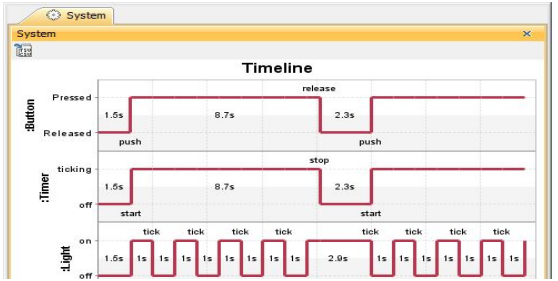

or Planning and Discussi
Purposes Only

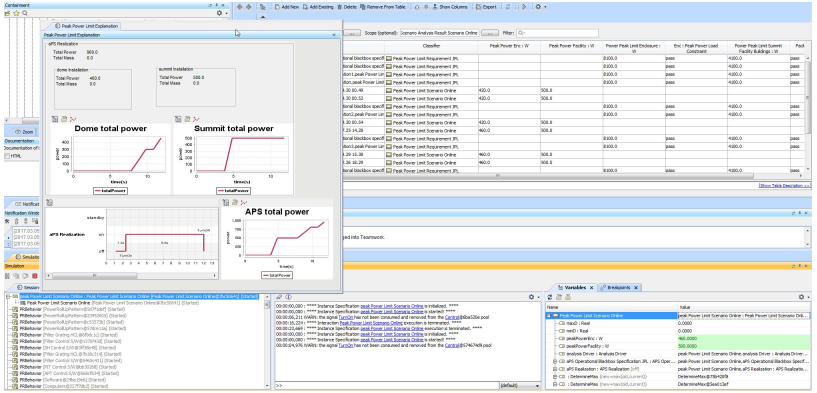
Use of signals sent over ports to simulate a message passing mechanism between components

Also across subsytems! (e.g., APS to M1CS)

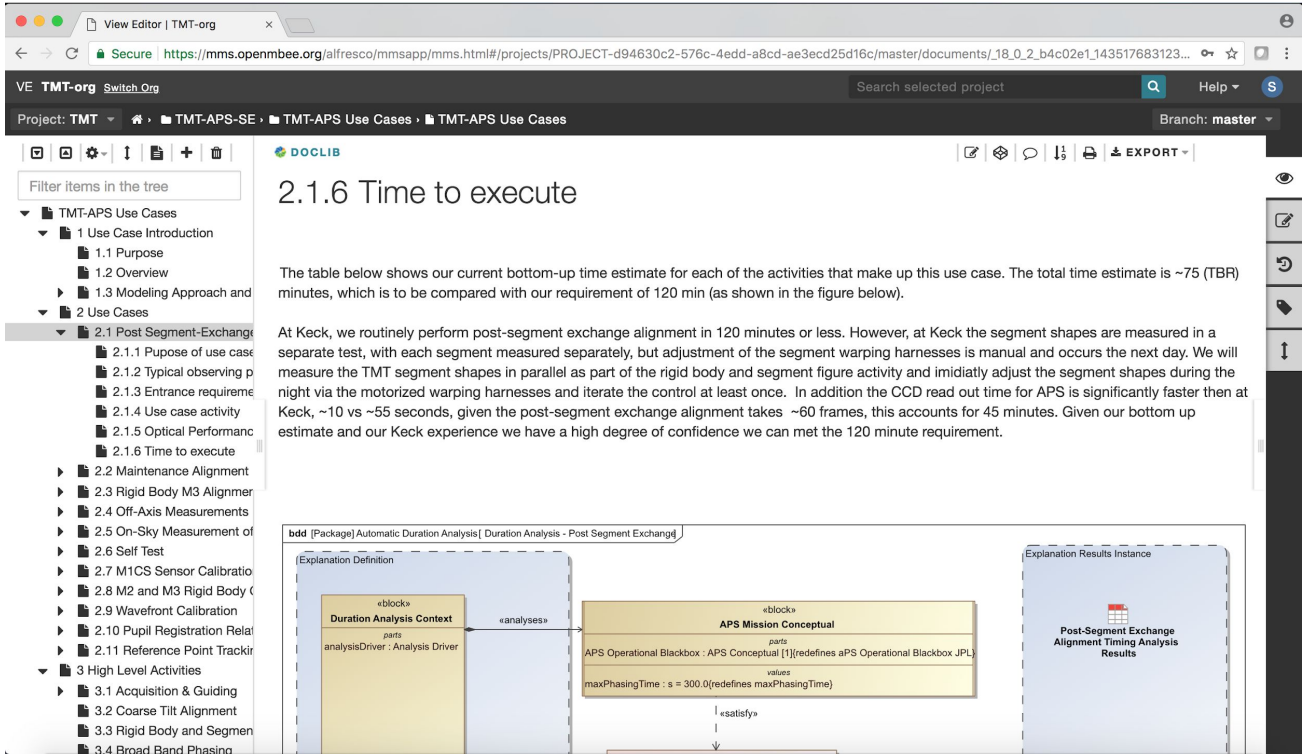
# Run Analyses

- Run a configured analysis with a simulation engine on the initial conditions to get the final conditions
- Produce the analyses declaratively, repeatably (in any system), without a single line of project-specific code -> reducing time and resources
- Produce the following views on final conditions
  - Table showing final analysis values (e.g. peak power) and the constraint's pass/fail status for each scenario
  - Timelines: state changes for components over time
  - Value profiles: total rolled up values over time



# Document & Report Generation with View Editor




* OpenMBEE / ViewEditor is open source, and available at https://www.openmbee.org

# Enabling Technology

## Bridging SE and people with models and data

fmi SysML v2 jupyter

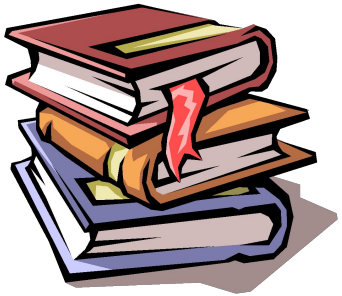
OpenMBEE

# Model Based Systems Engineering (MBSE)

- MBSE is the **formalized** application of modeling techniques to support system requirements, design, analysis, verification, validation and documentation activities
- MBSE expresses a system using a **Systems Modeling Language** (e.g. SysML, Modelica)
- MBSE is often applied with a **method** like Object Oriented System Engineering Method (OOSEM)

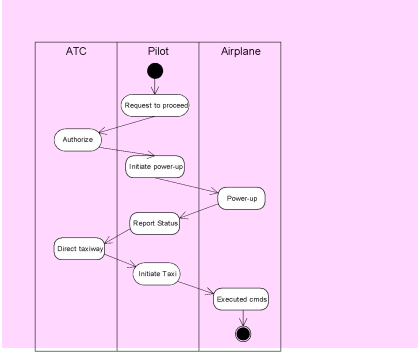
# Vision from early 2000's

**Present**



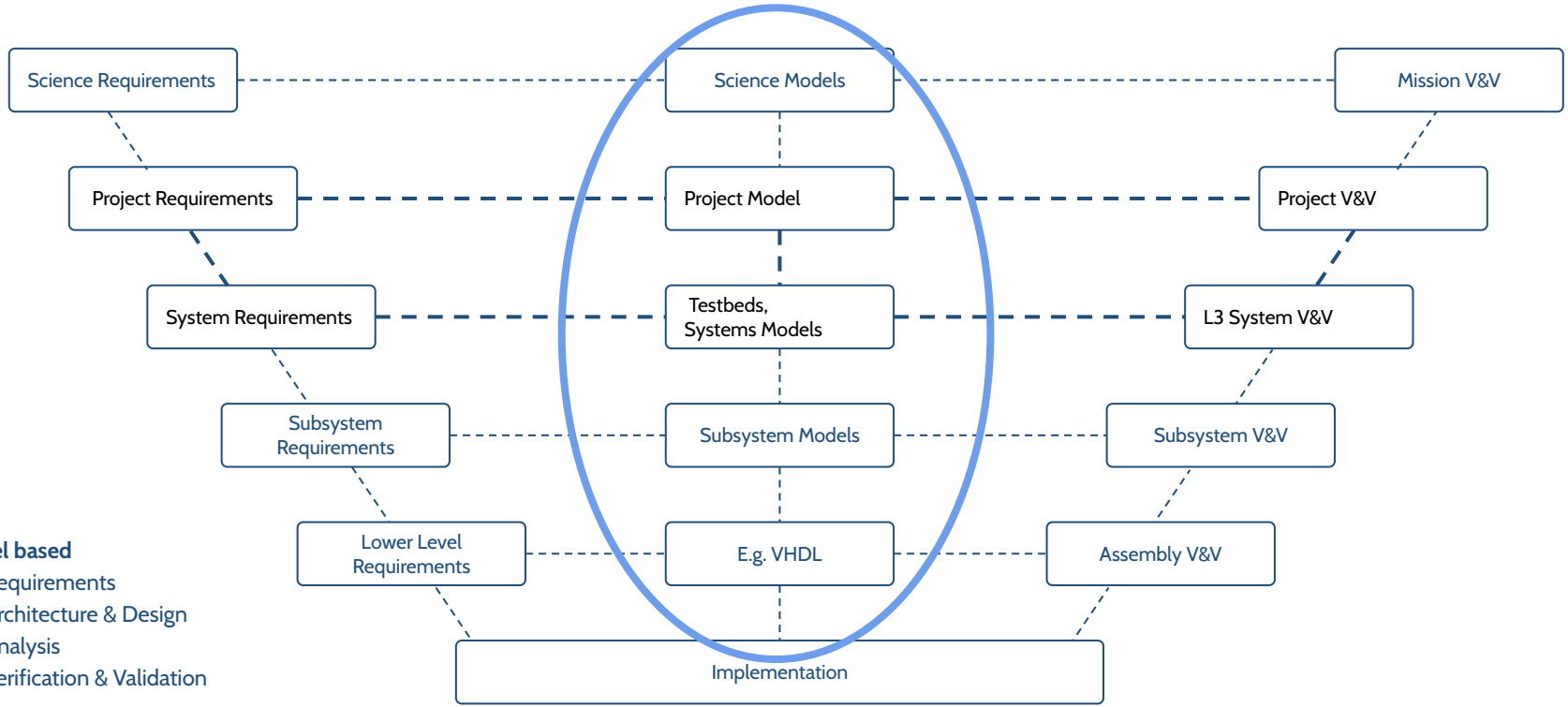
- Specifications
- Interface requirements
- System design
- Analysis & Trade-off

**Future**




Moving from Document centric to Model centric

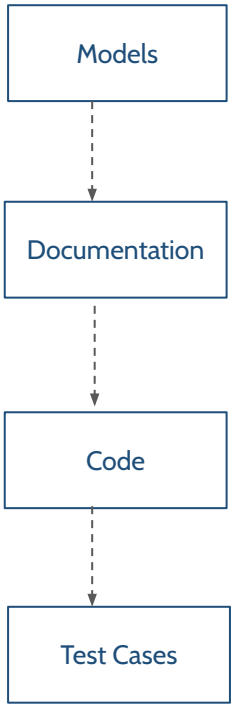
# Model Driven V

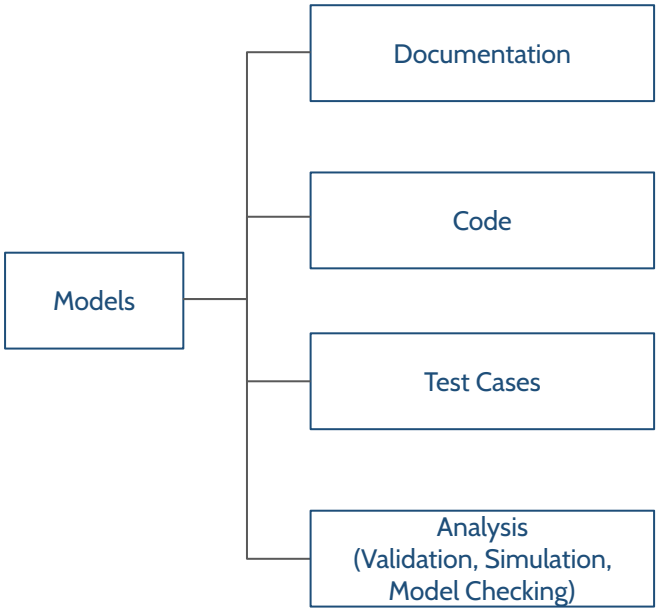



**Model based**

- Requirements
- Architecture & Design
- Analysis
- Verification & Validation

# Model Driven instead of Document Driven – from System to Software



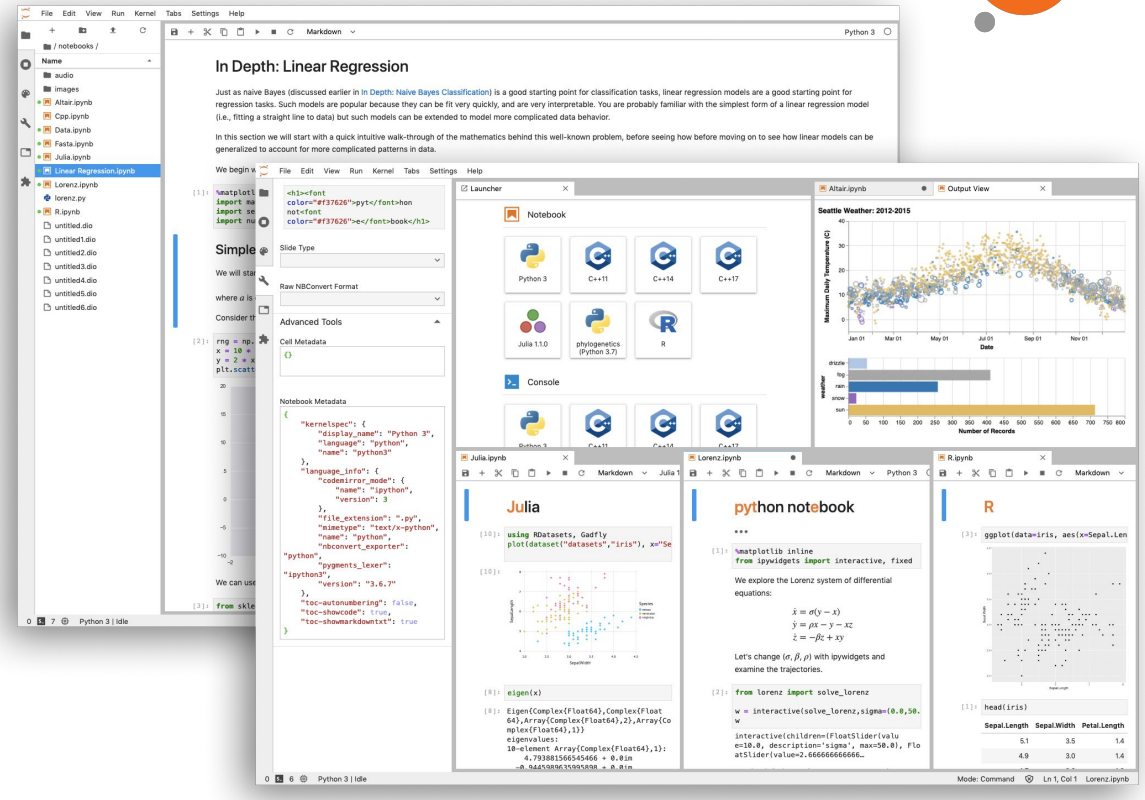




Semantic consistency across all transformed artifacts

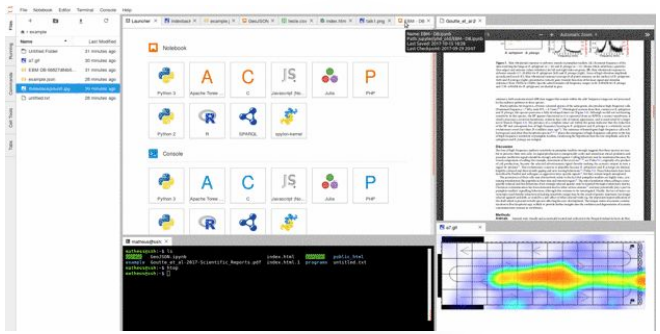
# Jupyter as Analysis and Visualization hub for Models

Interactive, exploratory, browser-based computing environment for:

- Engineering
- Data science
- Scientific computing
- ML/AI
- and so much more...



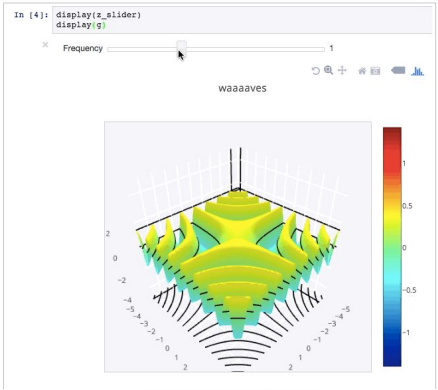
# JupyterLab Features



## Rich engineering narrative

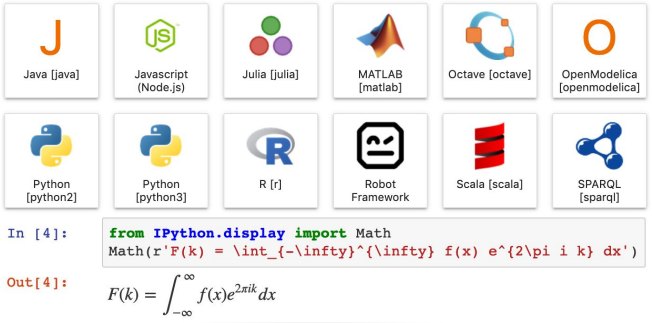
Feature rich and integrated computational environment

Notebooks as reproducible and executable reports




## Interactive & visual

Live code, narrative text, equations (LaTeX), images, visualizations, audio, video, widgets




## Widely popular & supported

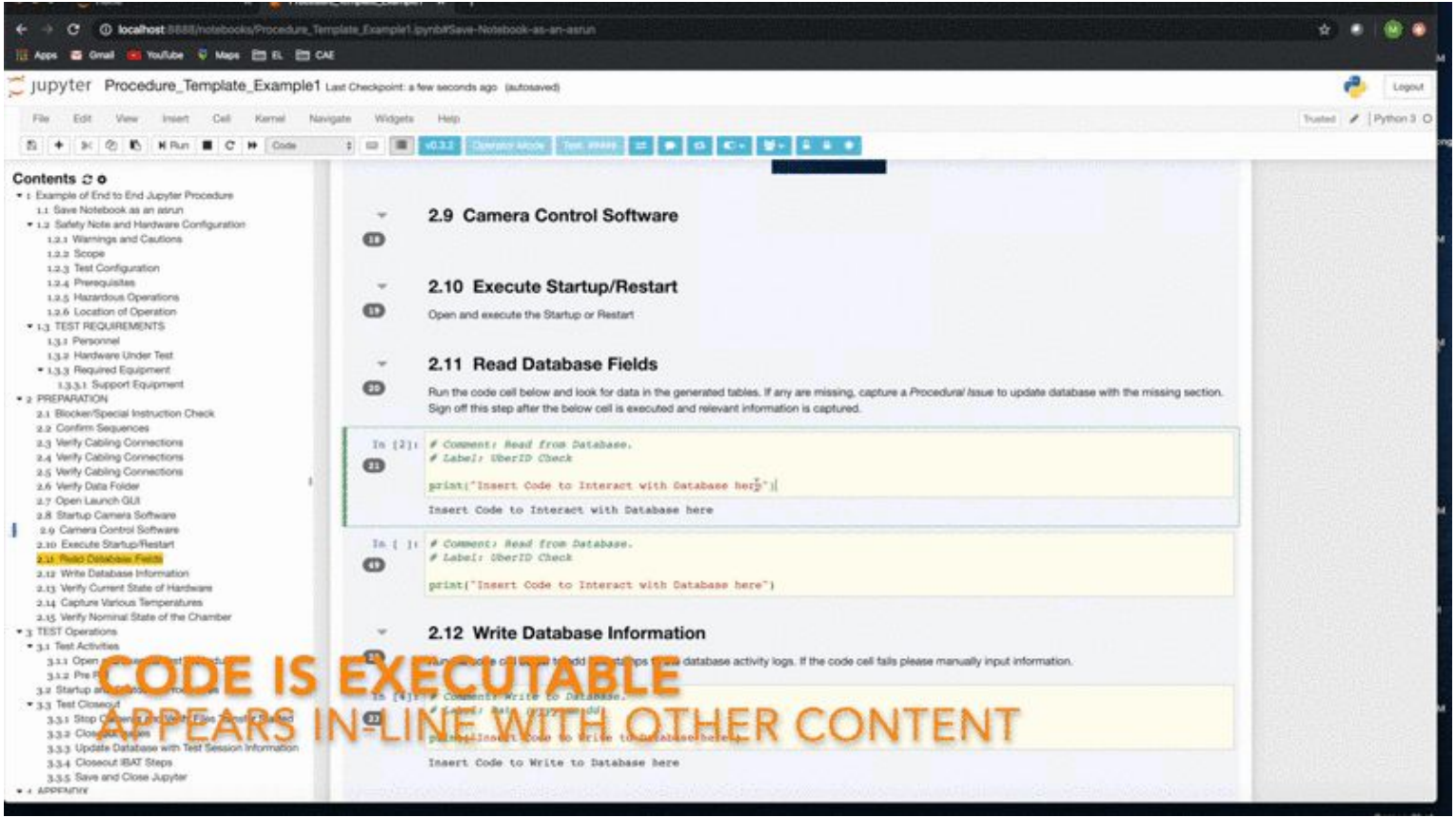
~100 programming languages supported

Millions of public notebooks on GitHub to find inspiration

# Robotic Arm on the Move

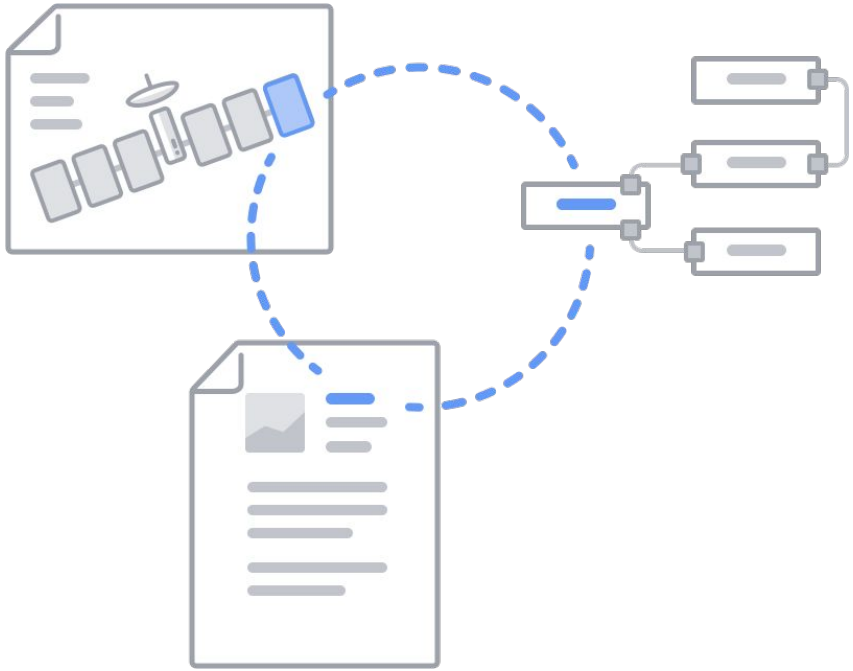
Source: https://mars.nasa.gov/resources/22739/nasas-mars-2020-rover-robotic-arm-is-on-the-move-time-lapse/

# Test Procedures



# Open Model-Based Engineering Environment

- OpenMBEE is a **community** for open source modeling software and models
  - Open source software activities
  - Open source models
  - Open source exchange of ideas
- Participants and adopters:
  JPL, Boeing, Lockheed Martin, OMG, NavAir, Ford, Stevens, Georgia Tech, ESO, ...
- > 400 members



Linked Data Documents with OpenMBEE

# Engineering Documents

## From Engineering Models to Engineering Documents

OpenMBEE


# Systems Engineers guide the concurrent collaborative design of complex technical systems

# Project teams are large

# A project starts simple

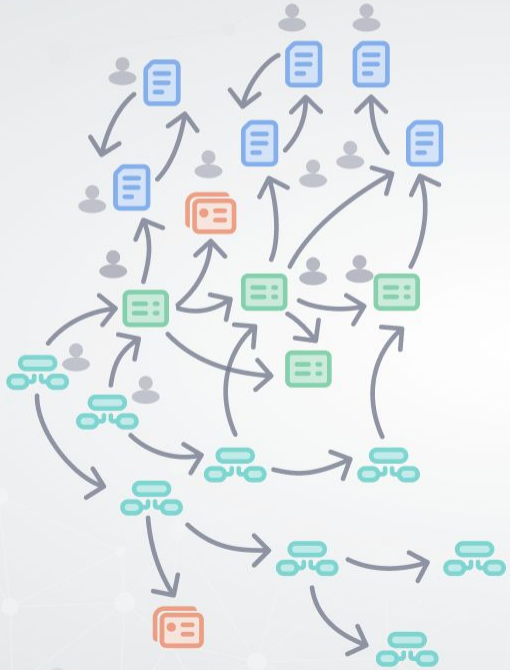
# Engineers iterate on their models

# Systems engineers enter this data into their rollup

# They add it to a document and get inputs from others

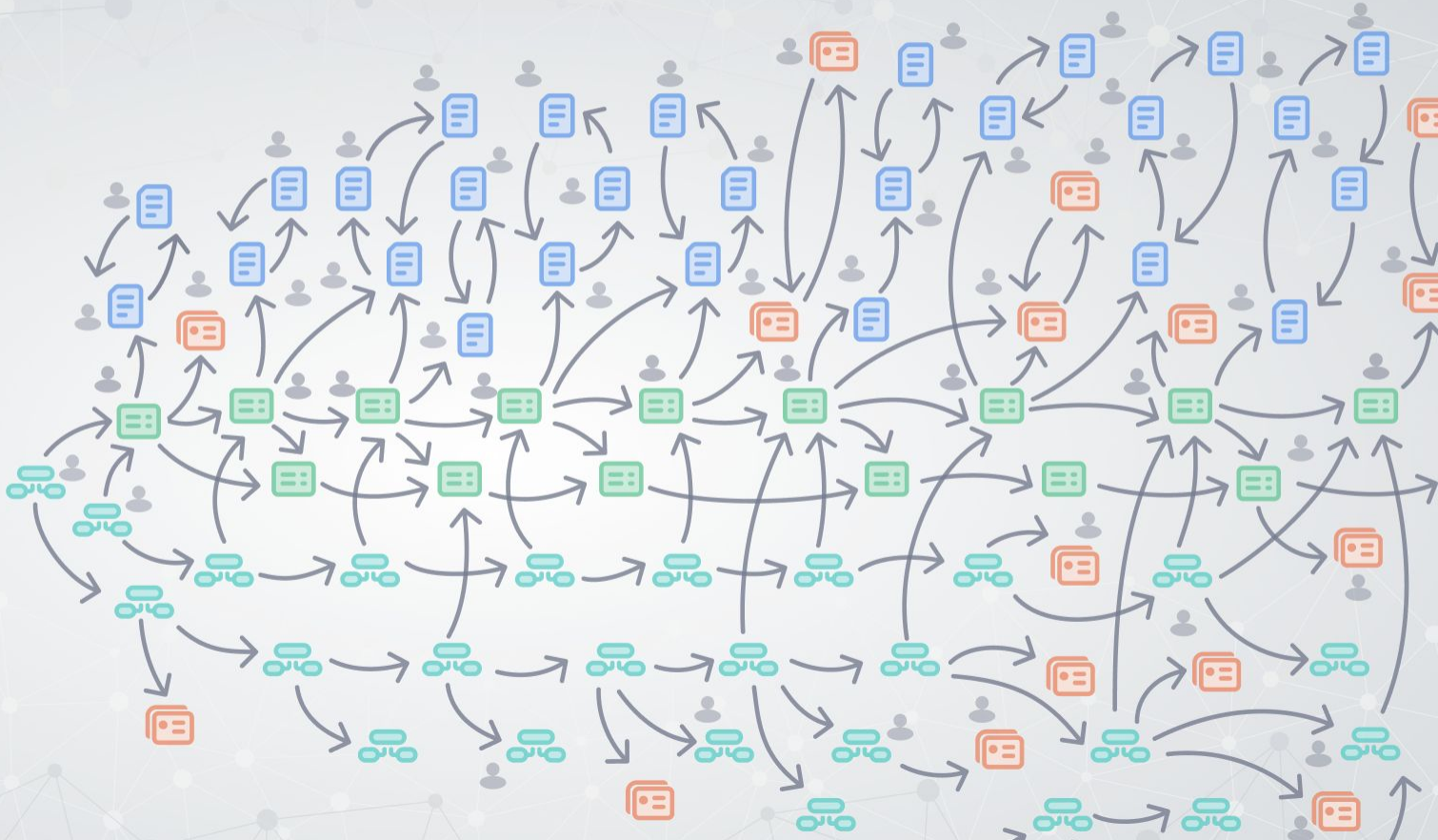
# And add it to a spreadsheet to track it over time

# It can get complicated quickly

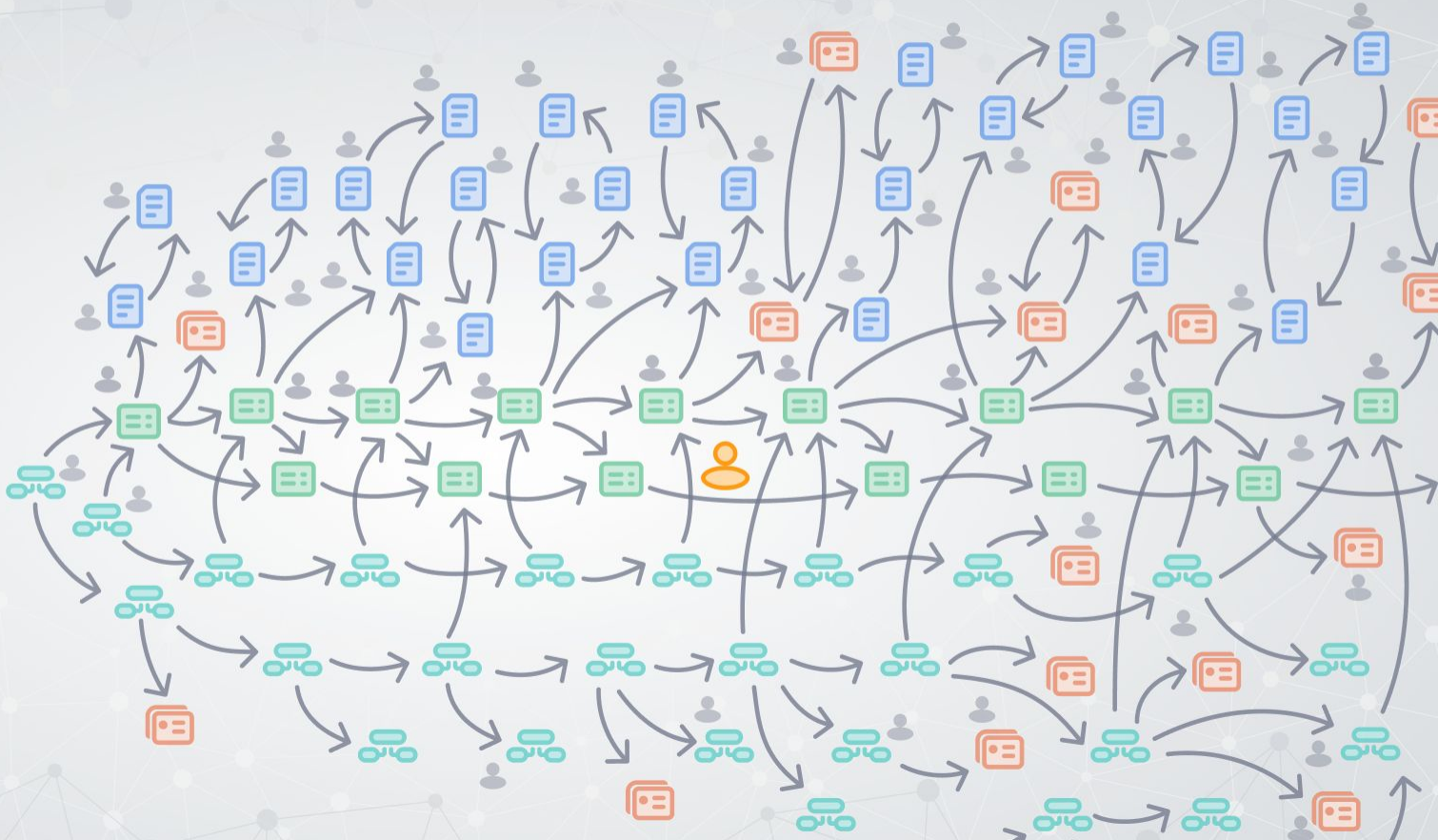


# It can get complicated quickly

# It can get complicated quickly



# And become overwhelming...



**Most of my job is data entry.**
**I want to do REAL ENGINEERING.**

JPL Systems Engineer

> **Using engineers as information janitors isn’t the best use of their skill.**

JPL Systems Engineer

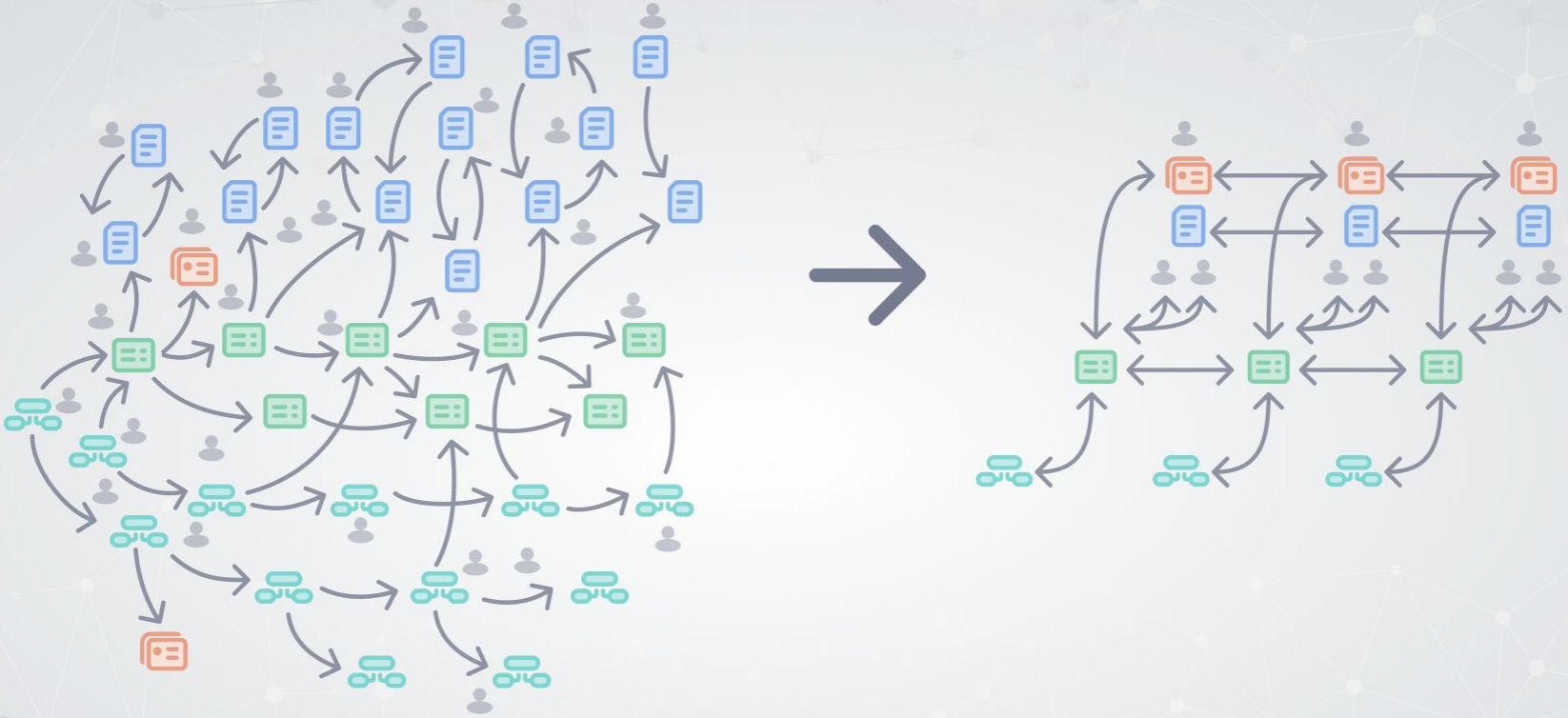
# Real engineering vs. overhead

Repetitive Data Entry

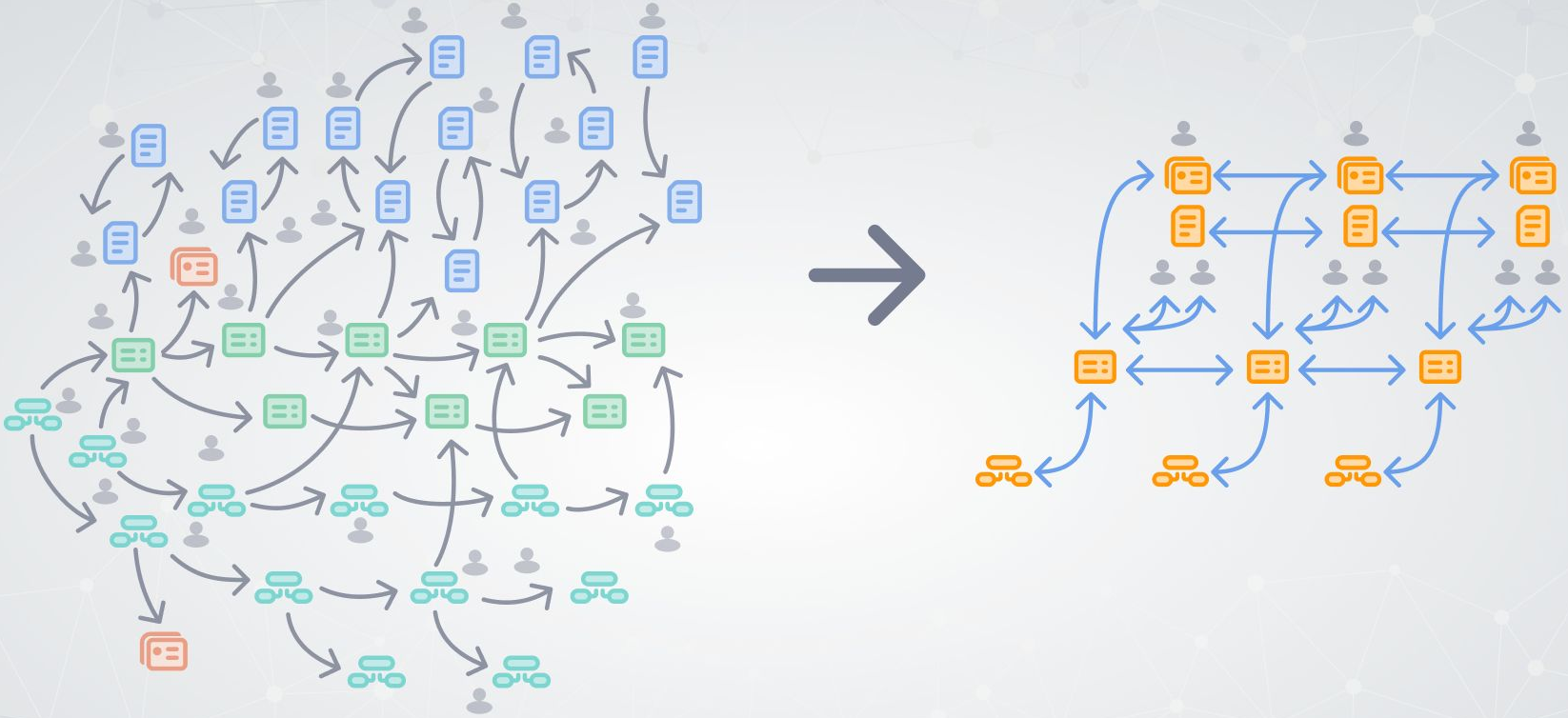
Version Confusion

Constant Searching

# A better way...



# A better way...



# Connected information, connected engineers

OpenMBEE

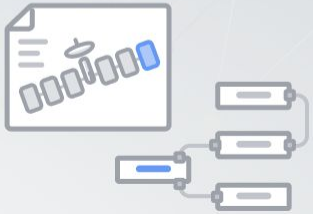
open source initiative

100% JPL
Core Components

TRL 8

Model Management System

View Editor & Platform for Model Analysis



Model Development Kits

Documentation and Training

Thirty Meter Telescope Model

OpenSE Cookbook

OpenMBEE

Thirty Meter Telescope Model

Model Management System

Authoritative Source

OpenSE Cookbook

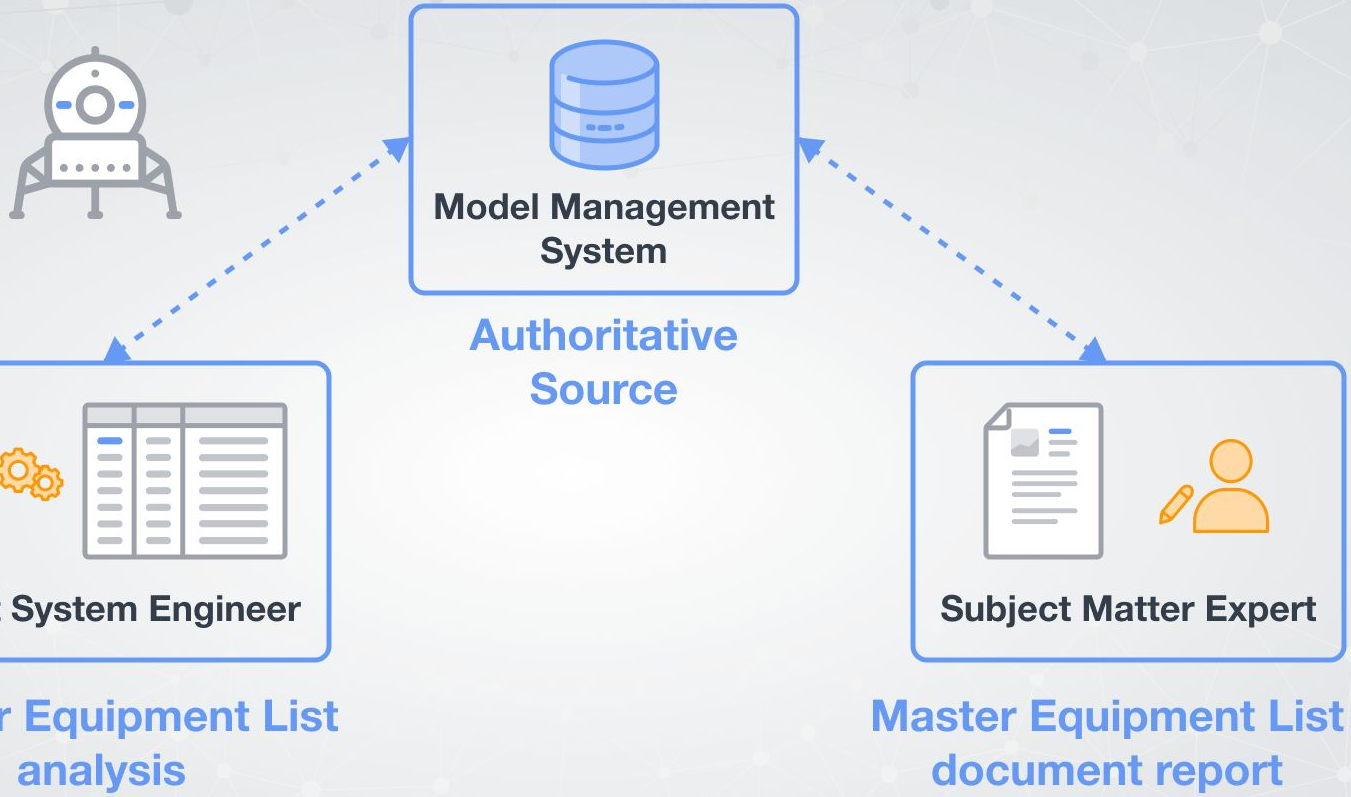
Model Development Kits

Engineering Modeling

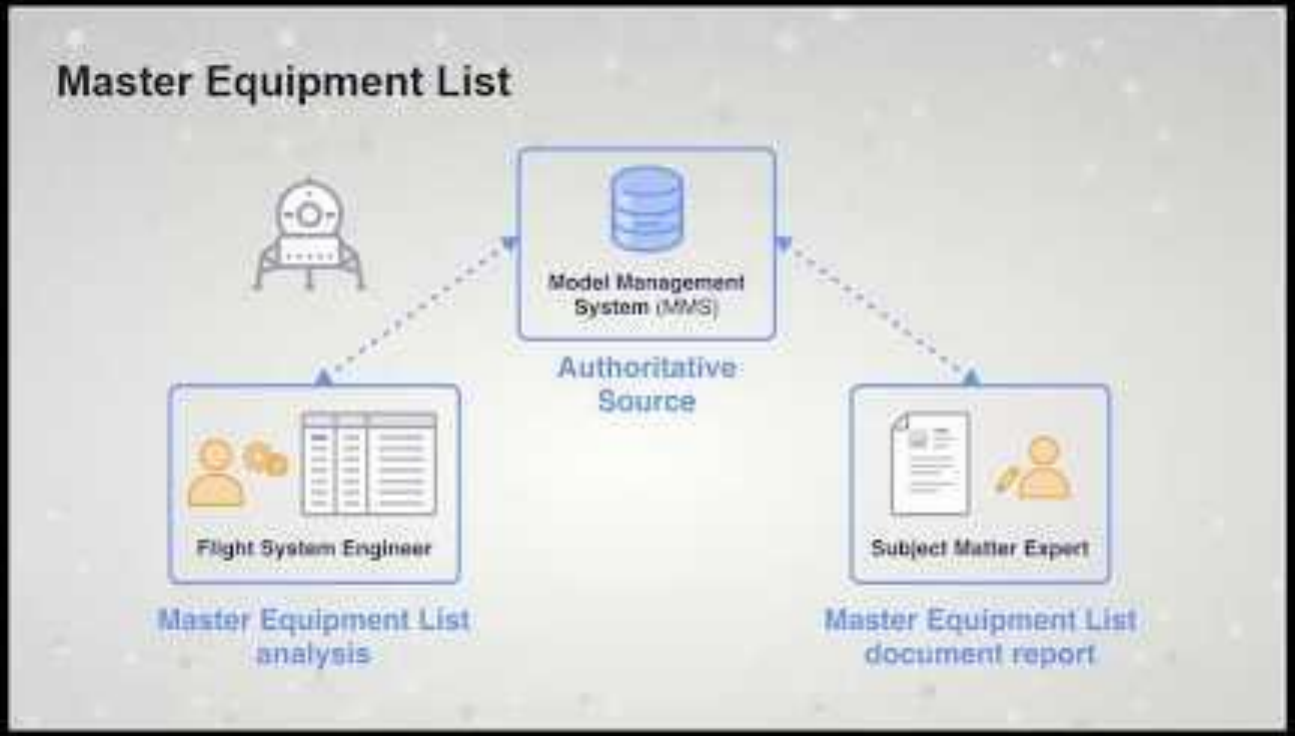
View Editor & Platform for Model Analysis

Document Authoring

# Master Equipment List



# Demo


Master Equipment List
Model Management System (MMS)
Authoritative Source
Flight System Engineer
Master Equipment List analysis
Subject Matter Expert
Master Equipment List document report

# OpenMBEE adopters and contributors

## NASA

Mars 2020

Europa Clipper

ARRM

Mars Sample Return

MAIA

SWOT

NASA Pathfinder

Europa Lander

## Science & Engineering

Thirty Meter Telescope

Japan Aerospace Exploration Agency

Stevens Institute of Technology

Systems Engineering Research Center

Georgia Tech Research Institute

Georgia Tech Aerospace Systems Design Laboratory

## Industry

Boeing

Ford Motor Company

Lockheed Martin Corporation

## Standards

Object Management Group

INCOSE

## Vendors

Dassault Systemes

IncQuery Labs

Capella

Intercax

Tom Sawyer

Phoenix Integrations

Maplesoft

# Flight Project Impact:  Reduction of Overhead


**Europa Clipper**

**100**

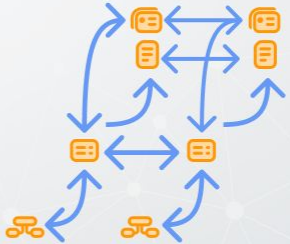
Concurrent users

**230+**

Documents and decision gate deliverables including

**445,000**

Connections between elements



# Flight Project Impact: Technical Rigor

**50**

Concurrent users

**Mars 2020**

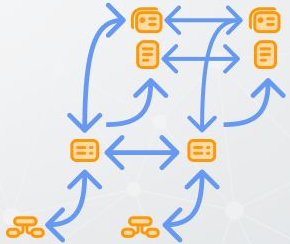
**90+**

Documents and decision gate deliverables including

- Reference Designator List
- Electrical Function List
- Electrical Functional Block Diagram

**180,000**

Connections between elements



# Aerospace Industry Impact: Enterprise Scalability

BOEING
Commercial Airplanes

**1,000**
Concurrent users

**50**
Programs

LOCKHEED MARTIN

**100**
Concurrent users

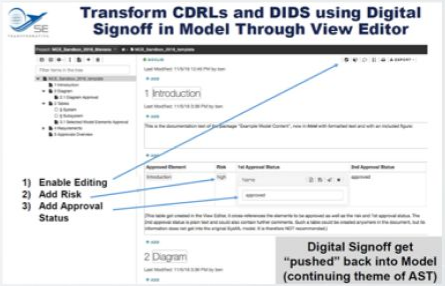
**50**
Projects

# Academia Impact: Inter-Organizational Collaboration

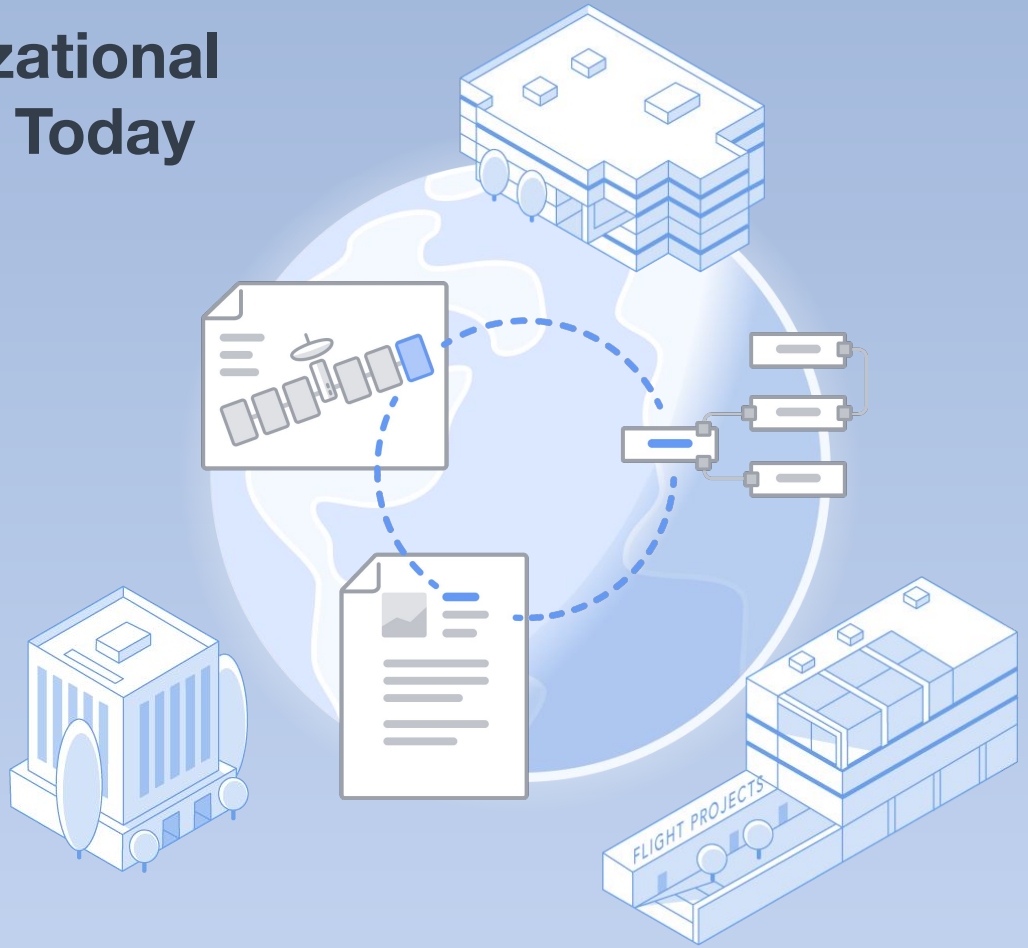
SYSTEMS ENGINEERING RESEARCH CENTER

## NAVAIR Model-Based Acquisition Strategy

- Surrogate Pilot RFP delivered to NAVAIR
- Data Item Descriptions (DIDs)
- Contract Data Requirements List (CDRL)

# Intra-Organizational Environment Today



# Inter-Organizational Concept

# Mars Sample Return Concept

# Global Engineering Ecosystem Vision

**Connected engineering information for a connected world**

# Questions?

# Backup

# OpenMBEE Vision

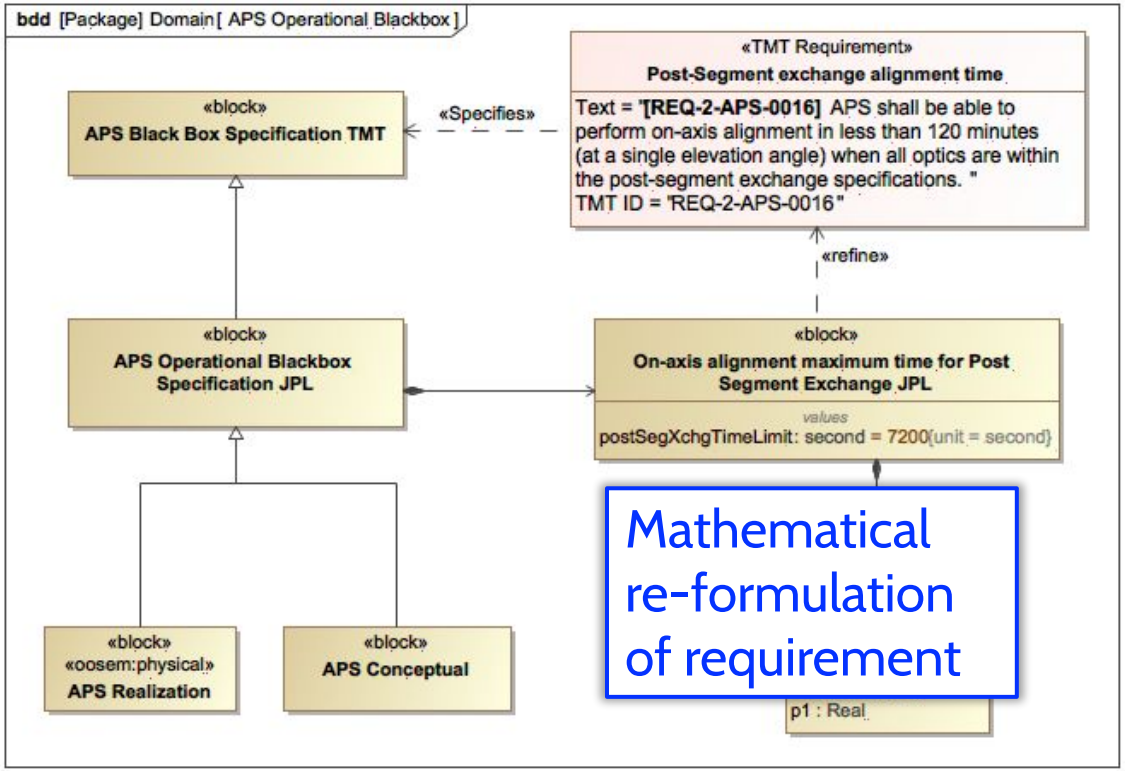
OpenMBEE

- Augment Jupyter's **multi-language analysis** capabilities with modeling and connected engineering
- Enable novel data-driven analyses with advanced capabilities, as a **service**
- Unlock value through **commoditization**
- **Standards** powered engineering **platform** using SysML v2 + API & Services, FMI

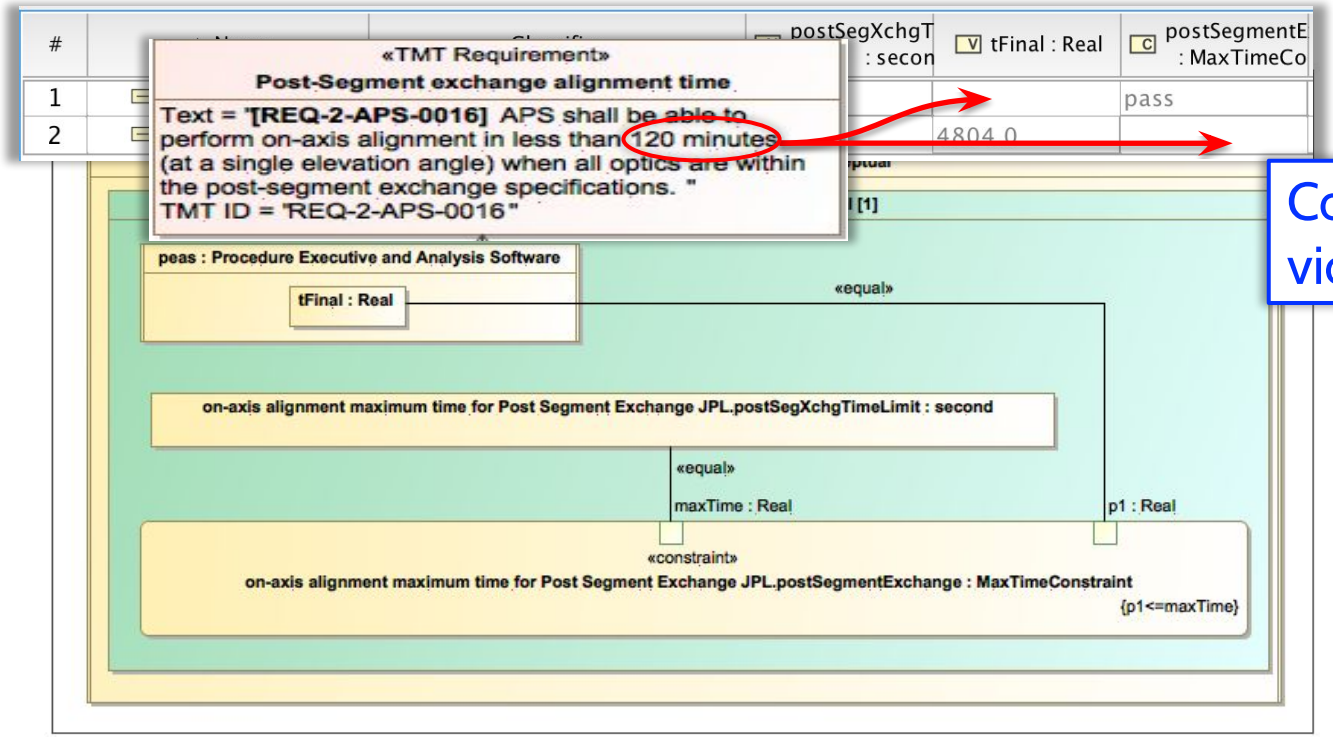
Global Engineering Ecosystem

**Join the community in making this vision into a reality @ openmbee.org**

# Formalizing Requirements



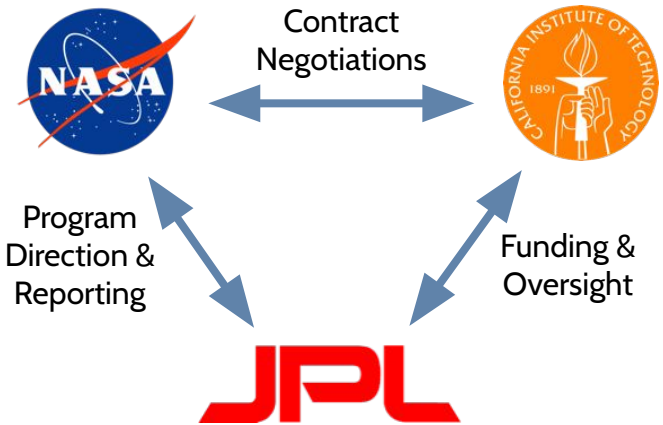
# Verifying Timing Requirements by Simulation

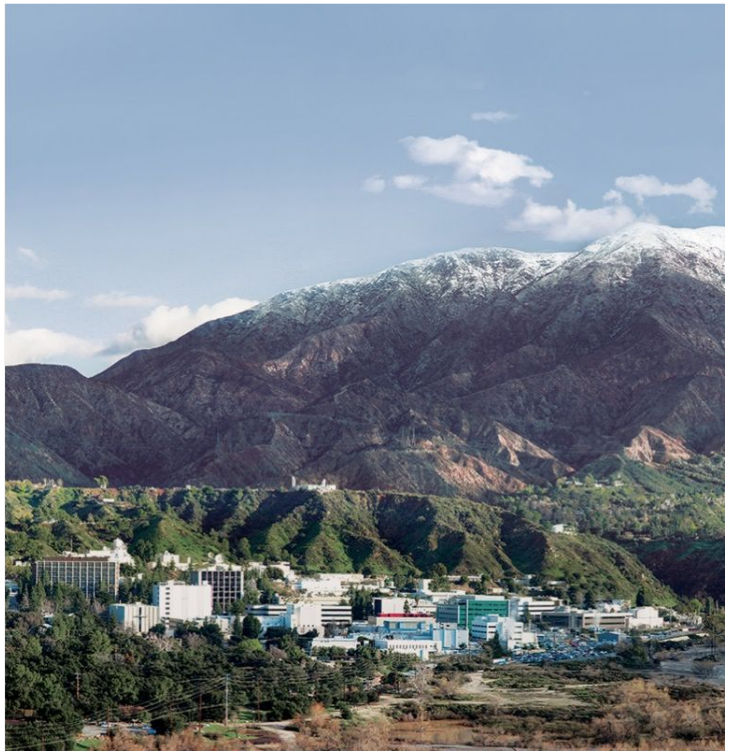



Constraint is either violated or not

# NASA Jet Propulsion Laboratory

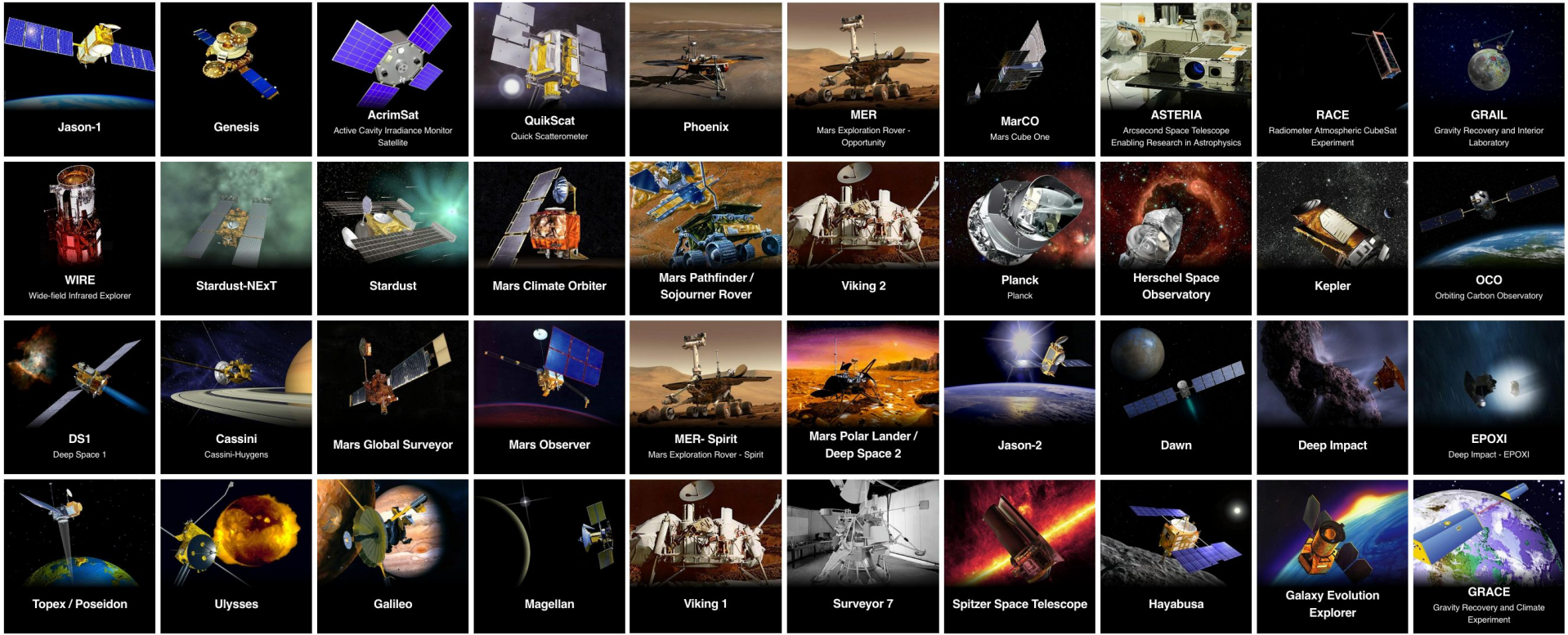
- Located in Pasadena, CA
- NASA-owned *"Federally-Funded Research and Development Center"*
- University-operated
- ~6,000 employees



JPL Campus in Pasadena, CA

# Past Missions




Source: https://www.jpl.nasa.gov/missions/

# Current Missions



Source: https://www.jpl.nasa.gov/missions/