ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns

July 18, 2016, Nerijus Jankevičius
No Magic Europe

Outline

- Current State of Practice and Need for MBSE
- TMT Application
- Executable System Engineering Method (ESEM)
- Demo video
- Summary

Authors

- Nerijus Jankevičius, nerijus@nomagic.com
- Product Manager @ No Magic Europe
- Leads the development of MBSE tools and solutions since 1997
- OMG member, co-author of UML and SysML languages

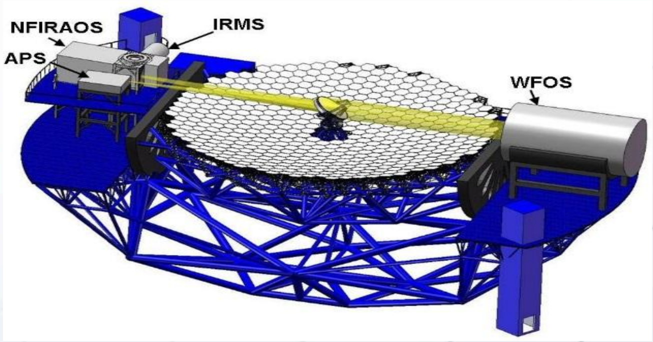
Robert Karban is a senior systems architect at the Jet Propulsion lab in the Systems Engineering and Formulation Division. Robert works in the domain of Model Based Systems Engineering (MBSE) as the task lead on providing a Model Based Engineering Environment (MBEE) for projects and applies modeling in the Thirty Meter Telescope project.

Maged Elaasar is a senior software architect at the Jet Propulsion Laboratory in the Systems Engineering and Formulation Division.

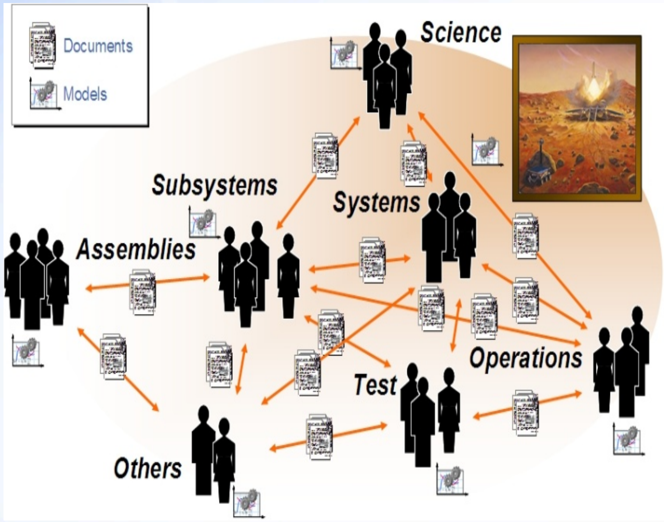
Thirty Meter Telescope (TMT)

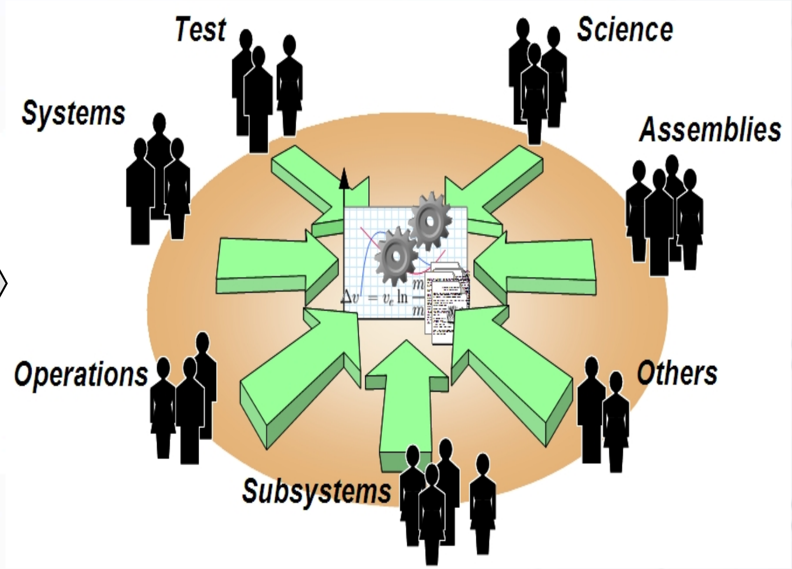
www.tmt.org

- Developed by TMT International Observatory (TIO)
  - JPL participates in several subsystems of TMT
  - APS (and AO) team uses MBSE to analyze requirements, produce design, and perform analysis
- Alignment and Phasing System (APS)
  - Sensor responsible for measuring the pre-adaptive optics wavefront quality



Current Practice to Future Practice





Today: Standalone models related through documents

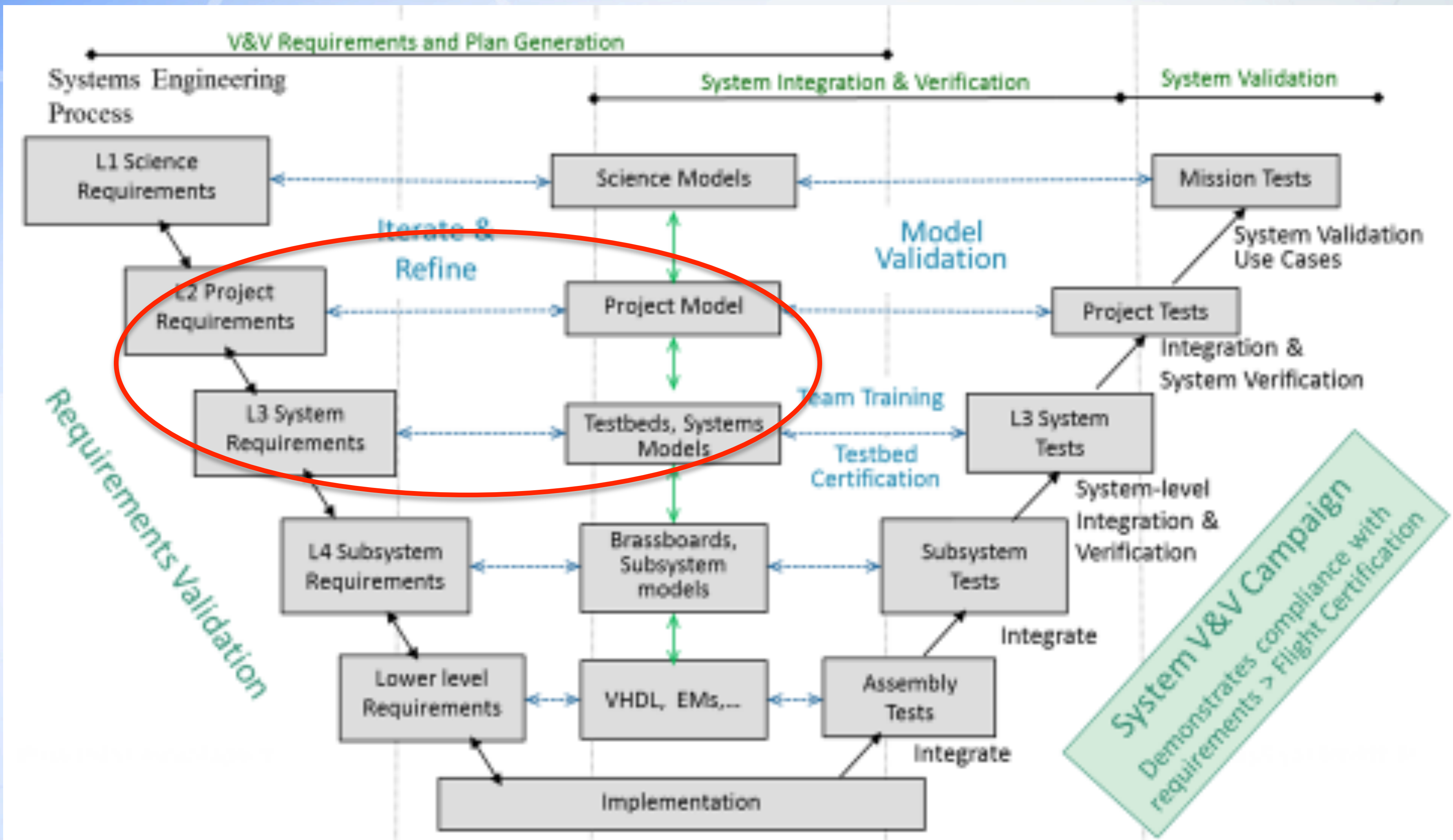
Future: Shared system model with multiple views, and connected to discipline models

Source : MBSE 101 by Elyse Fosse

Model Based Systems Engineering

- 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 (SysML), a profile of UML
- MBSE is often applied with a method like Object Oriented System Engineering Method (OOSEM)
- OOSEM maps onto the ISO systems engineering process and integrates top-down (**functional decomposition**) approach with **model-based** approach

Model-based V-Process at JPL



OpenMBEE

https://github.com/Open-MBEE

TMT Thirty Meter Telescope

- Provides a platform for modeling that integrates JPL's mission environment OpenCAE, by incorporating a model repository that can be accessed for example with a rich SysML desktop client (MagicDraw) and a light-weight web-based client (ViewEditor)
- The model repository provides the following features:
  - Basic Infrastructure for Version, Workflow, Access Control
  - Flexibility of content
  - Support for Web Applications and Web-based API access
  - Multi-tool and multi-repository integration across engineering and management disciplines

Systems Analysis

- Carry out quantitative assessments of systems in order to select and/or update the most efficient system architecture and to generate derived engineering data.
- System analysis provides a rigorous approach to technical decision-making. It is used to perform trade-off studies, and includes modeling and simulation, cost analysis, technical risks analysis, and effectiveness analysis.

Requirements Verification

- A kind of systems analysis that assesses whether a system design meets the objectives and satisfies the constraints that are implied by the system requirements

Proposed Approach

- Next phase of system modeling emphasizes executable models to enhance understanding , precision, and verification of requirements
- Executable Systems Engineering Method (**ESEM**) augments the OOSEM activities by enabling executable models
  - ESEM produces executable SysML models that verify requirements
  - Includes a set of analysis patterns that are specified with various SysML structural, behavioral and parametric diagrams
  - Also enables integration of supplier/customer models

Thirty Meter Telescope

TMT MBSE Objectives

- Use MBSE to define **executable SysML model** that captures requirements, operational scenarios (use cases), system decomposition, relationships and between subsystems, etc.
- Use the model to **analyze the system design** for
  - power consumption, mass, and duration
- Produce **engineering documents**
  - Requirement Flow Down Document
  - Operational Scenario Document
  - Design Description Document
  - Interface Control Documents
- Uses **standard languages and techniques where practical to** avoid custom software development

Typical Analysis Activities Using ESEM

- Capture **operational use cases** with estimated durations of actions, e.g.
  - Post segment-exchange alignment: requirement: 2h; CBE 1h19m
- Capture **power and mass characteristics** of components
- Identify **involved subsystems**, e.g. Telescope Control System (TCS), M1 Control System (M1CS)
- Identify **interfaces and interactions** among subsystems
- **Analyze** associated scenarios
- Automatically **verify system requirements** are met
- Derive requirements for TMT subsystems
- Develop/refine timing requirements for algorithms, internal and external interface commands

MagicDraw
Cameo Simulation Toolkit

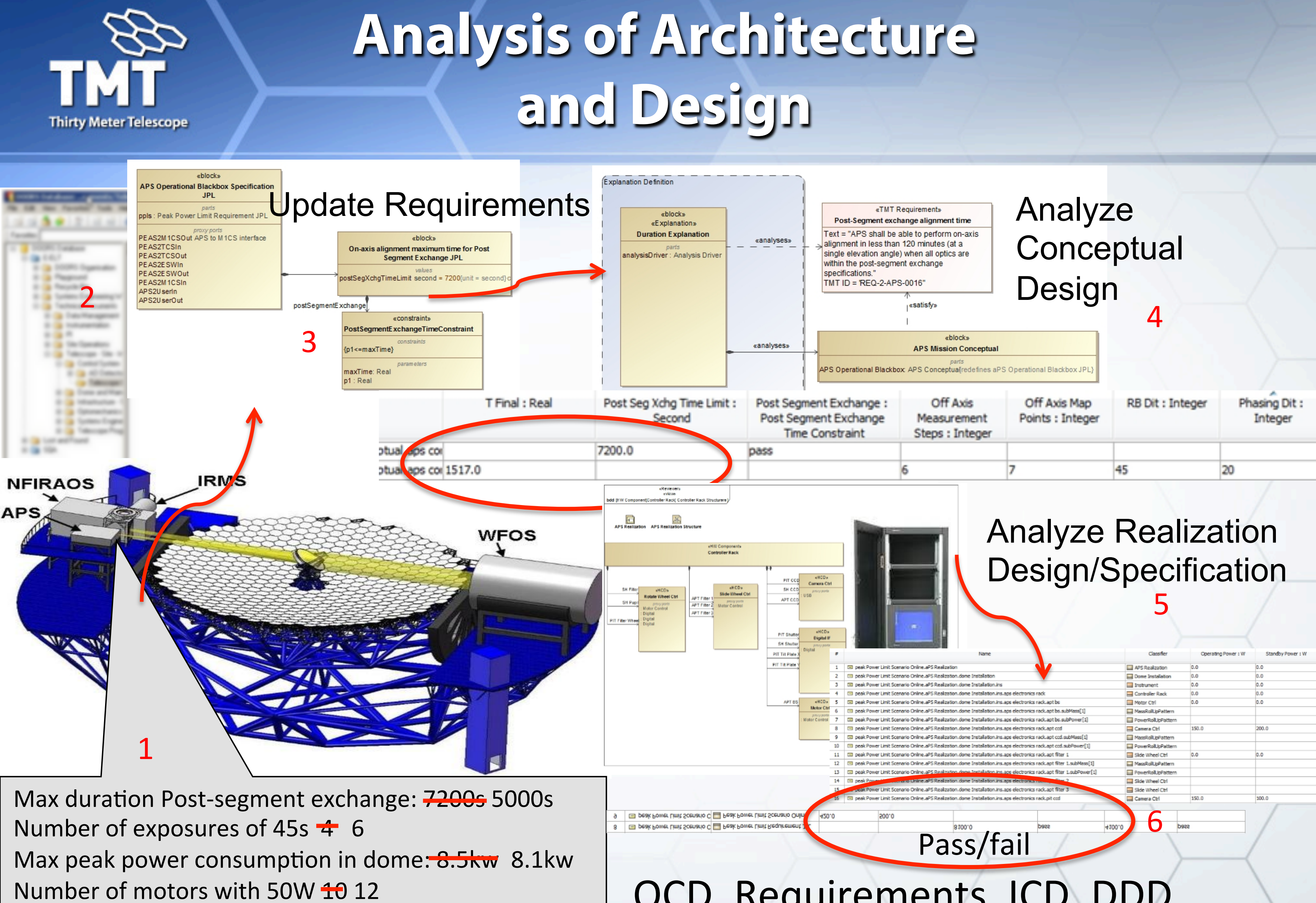
Thirty Meter Telescope

- Model execution framework and infrastructure:
  - Model debugging and animation environment
  - Pluggable engines, languages and evaluators
  - User Interface prototyping support
  - Model driven configs and test cases

- The standard based model execution of:
  - Activities (OMG fUML standard)
  - Composite structures (OMG PSCS)
  - Statemachines (W3C SCXML standard)
  - Actions/scripts (JSR223 standard)
  - Parametrics (OMG SysML standard)
  - Sequence diagrams (OMG UML Testing Profile)

OBJECT MANAGEMENT GROUP

Analysis of Architecture and Design



OCD, Requirements, ICD, DDD

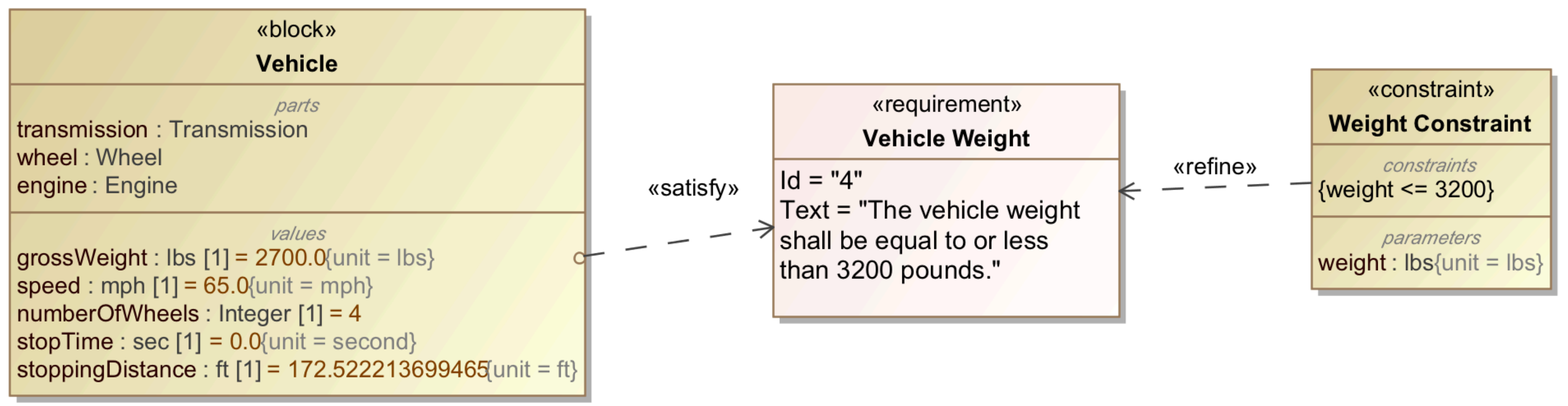
Executable System Engineering Method (ESEM)

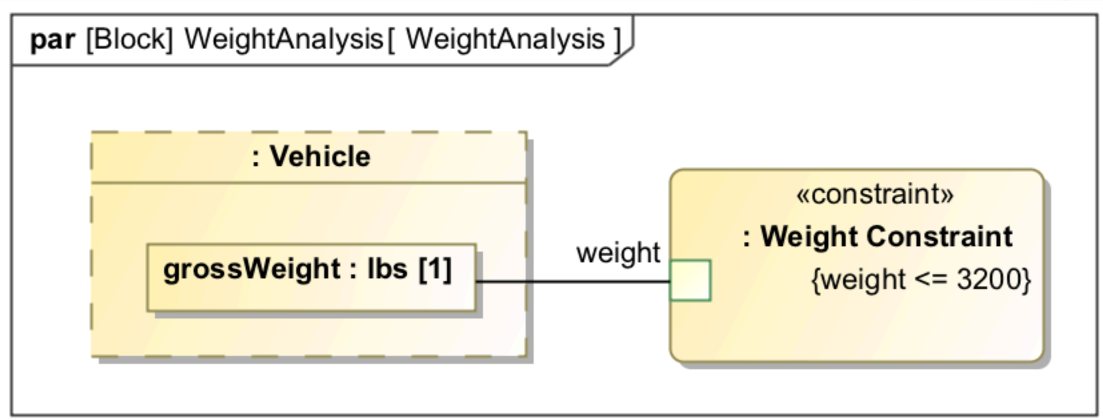
- Step 1: Formalize Requirements
- Step 2: Specify Design
- Step 3: Characterize Components
- Step 4: Specify Analysis Context
- Step 5: Specify Operational Scenarios
- Step 6: Specify Analysis Configurations
- Step 7: Run Analysis

Step 1: Formalize Requirements

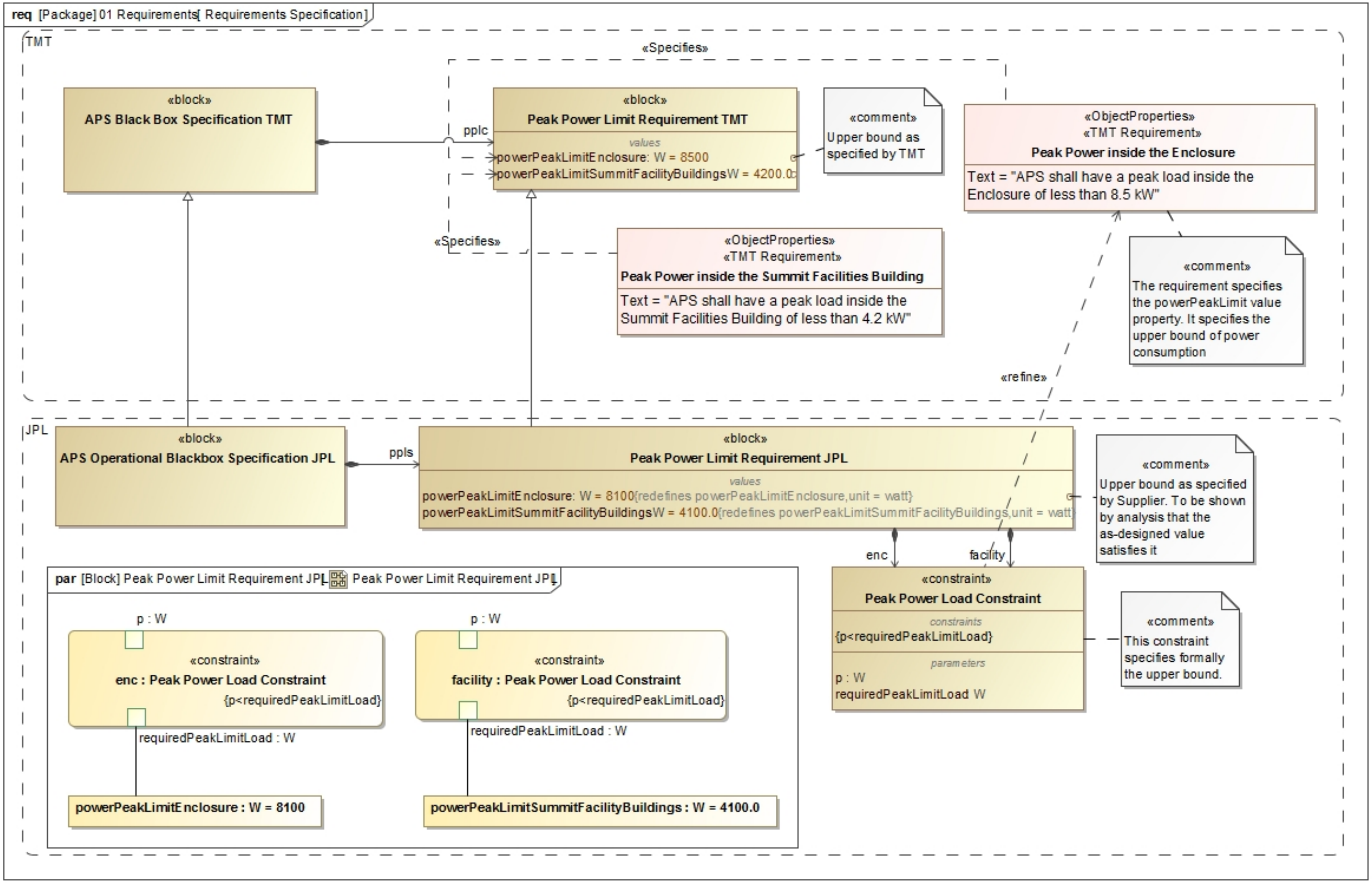
- Requirement Pattern
  - Customer Side
    - Define the textual requirement with a Requirement
    - Optionally define a design black box specification with a Block with relevant value properties
    - Optionally refine the Requirement with a Constraint Block on the black box design Block
  - Supplier Side
    - Define a design black box specification with a Block (that refines the customer's black box Block if any and provides tighter property values)
    - Refine the textual Requirement by a Constraint Block (if not already defined by the customer)

Satisfy, refine, bind





Step 1: Formalize Requirements

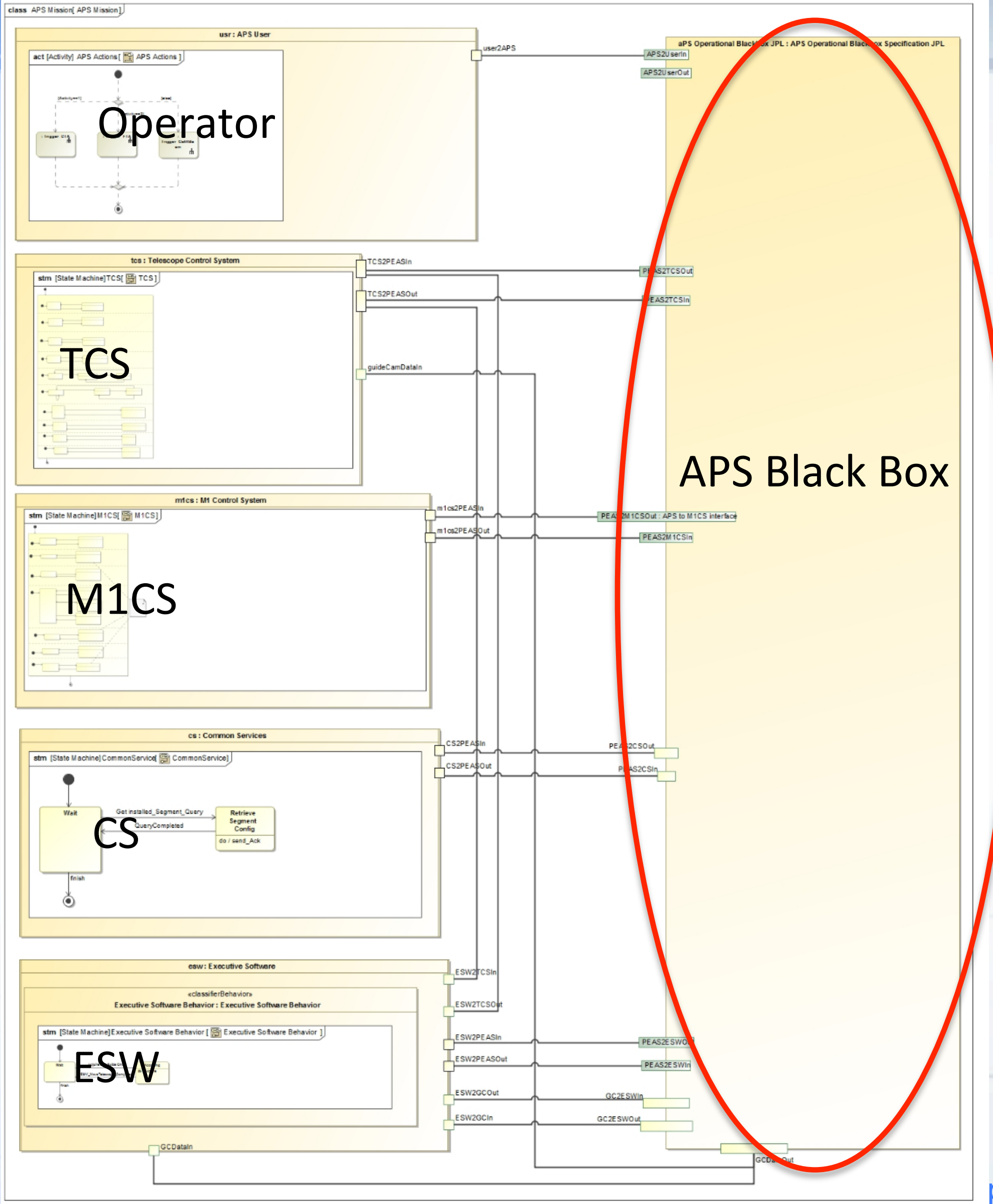


Step 2: Specify Design

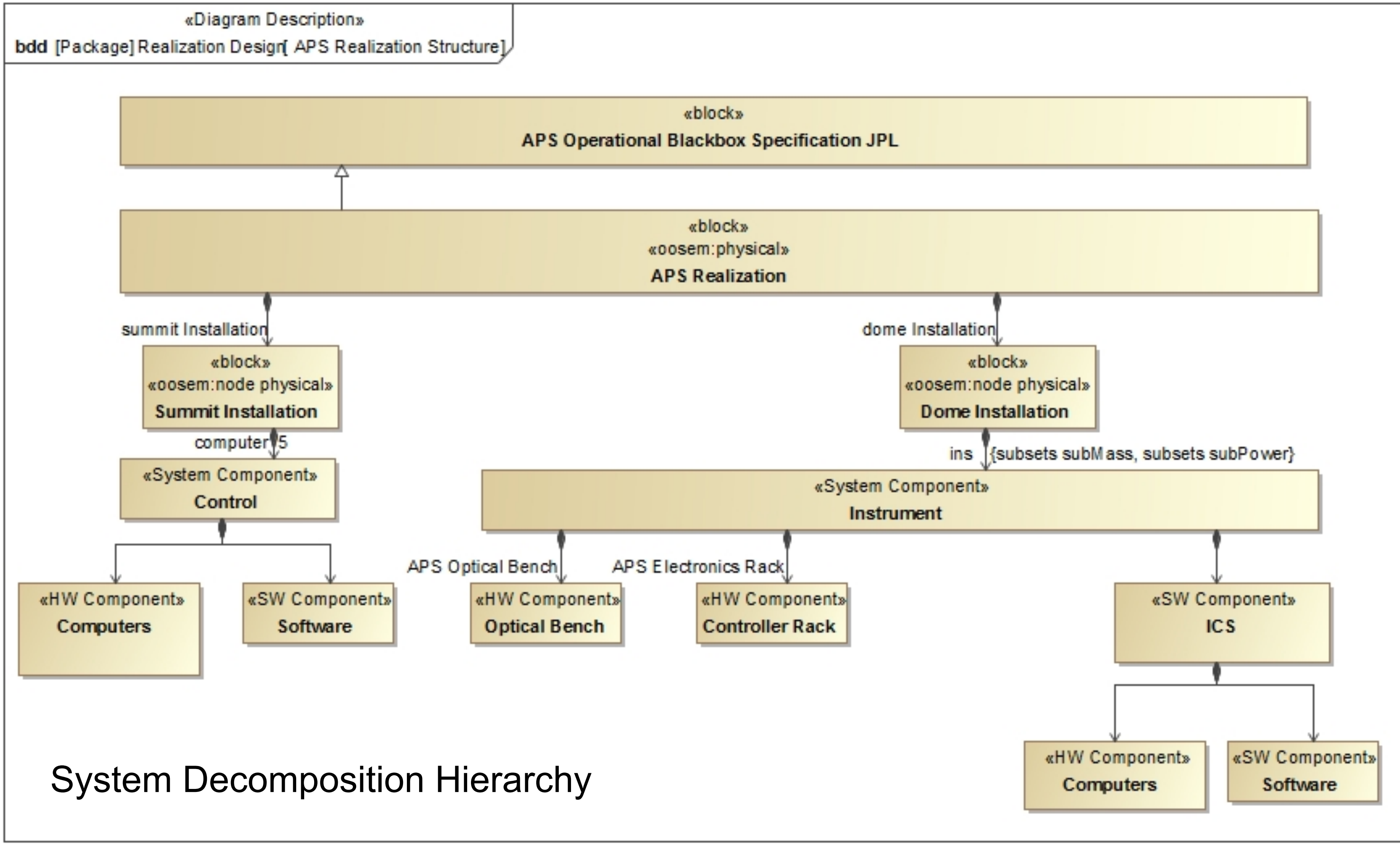
- Follow OOSEM to define two white box specifications that specialize the black box specification
  - Conceptual Specification
  - Realization Specification
- Decompose the white box designs into Blocks representing the subsystems

Black Box Level

Project level components communicate with APS black box block

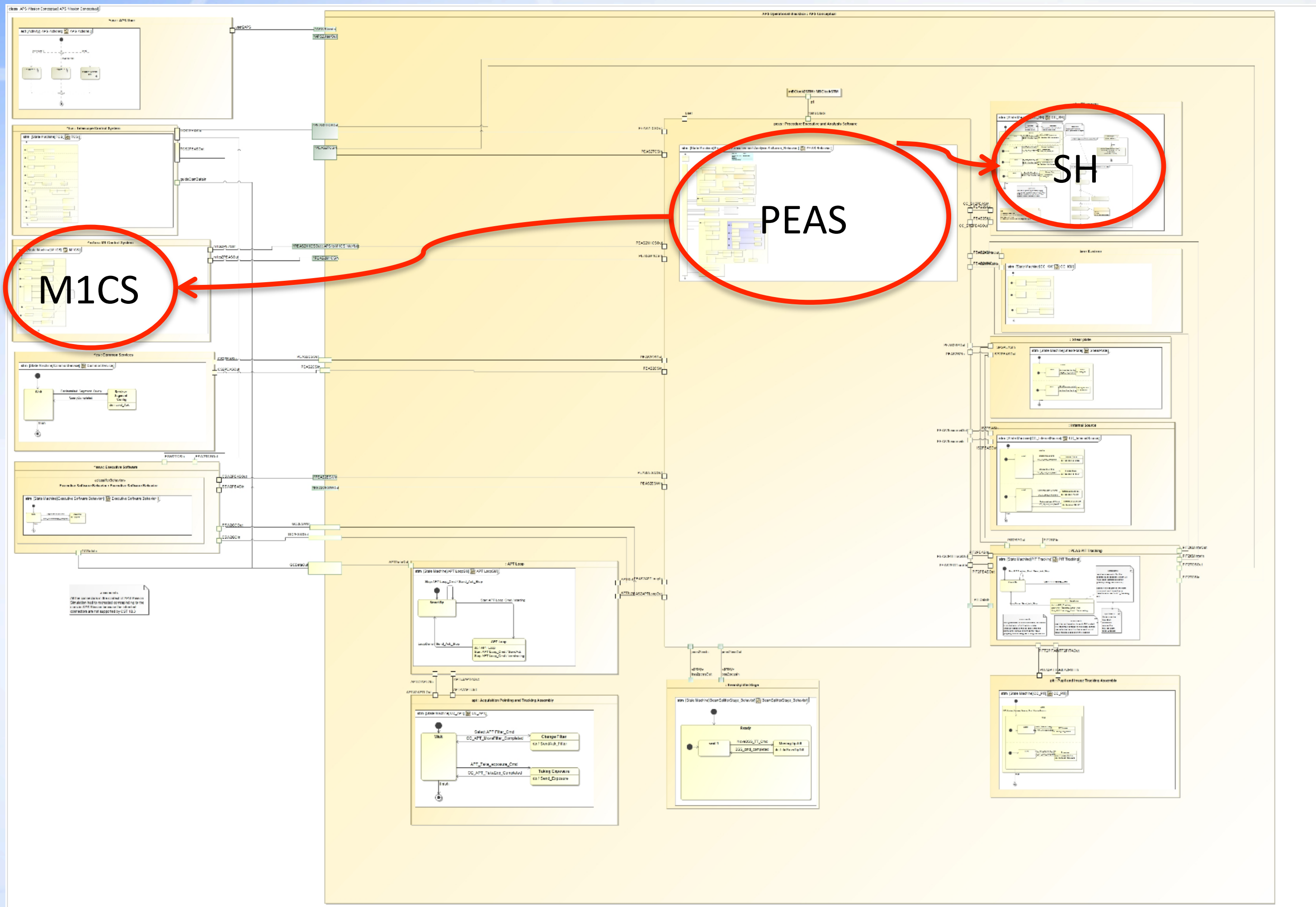


Step 2: Realization



System Decomposition Hierarchy

Step 2: Conceptual Model



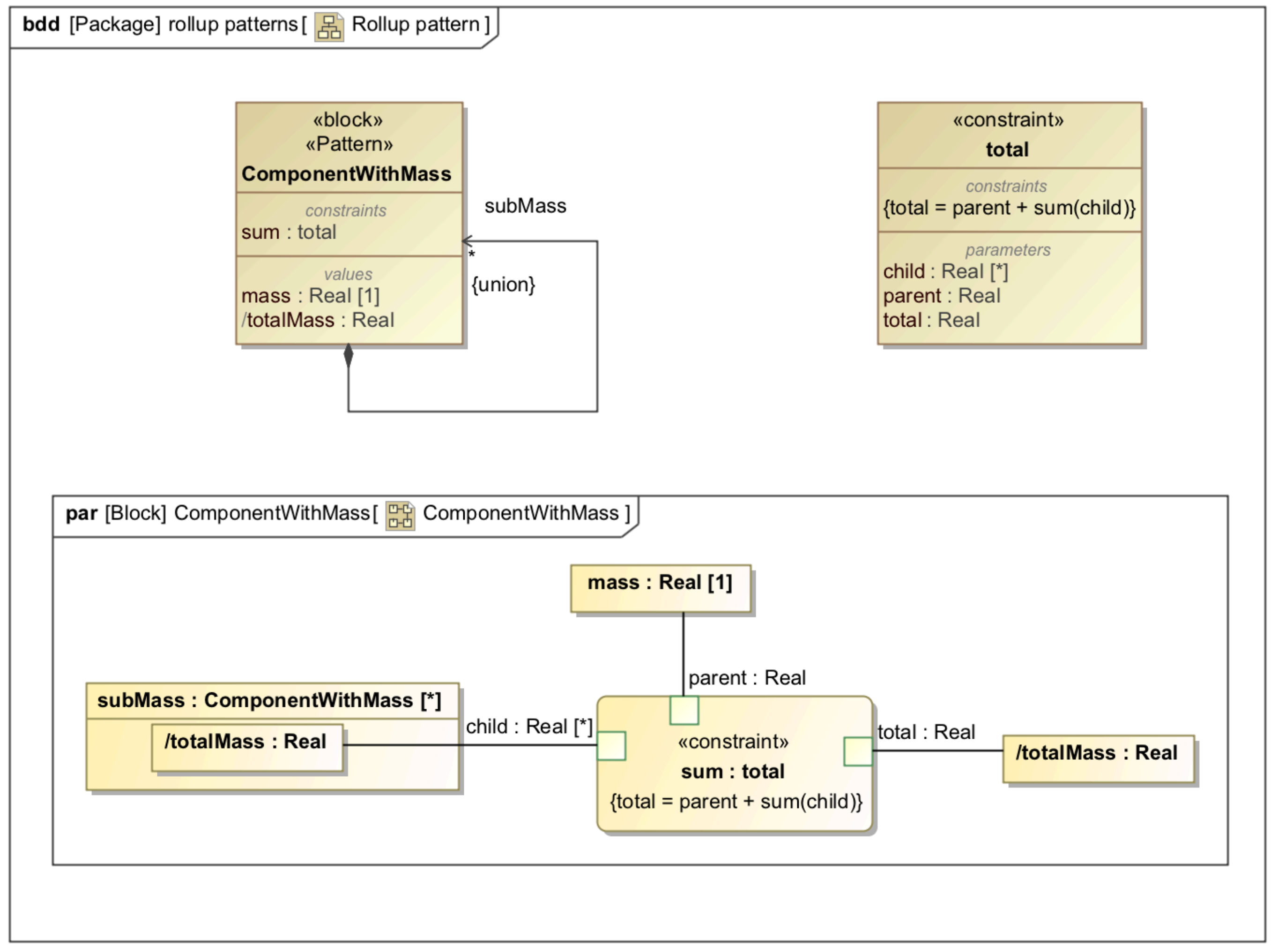
Communication between state machine specified components over ports

nowledged.

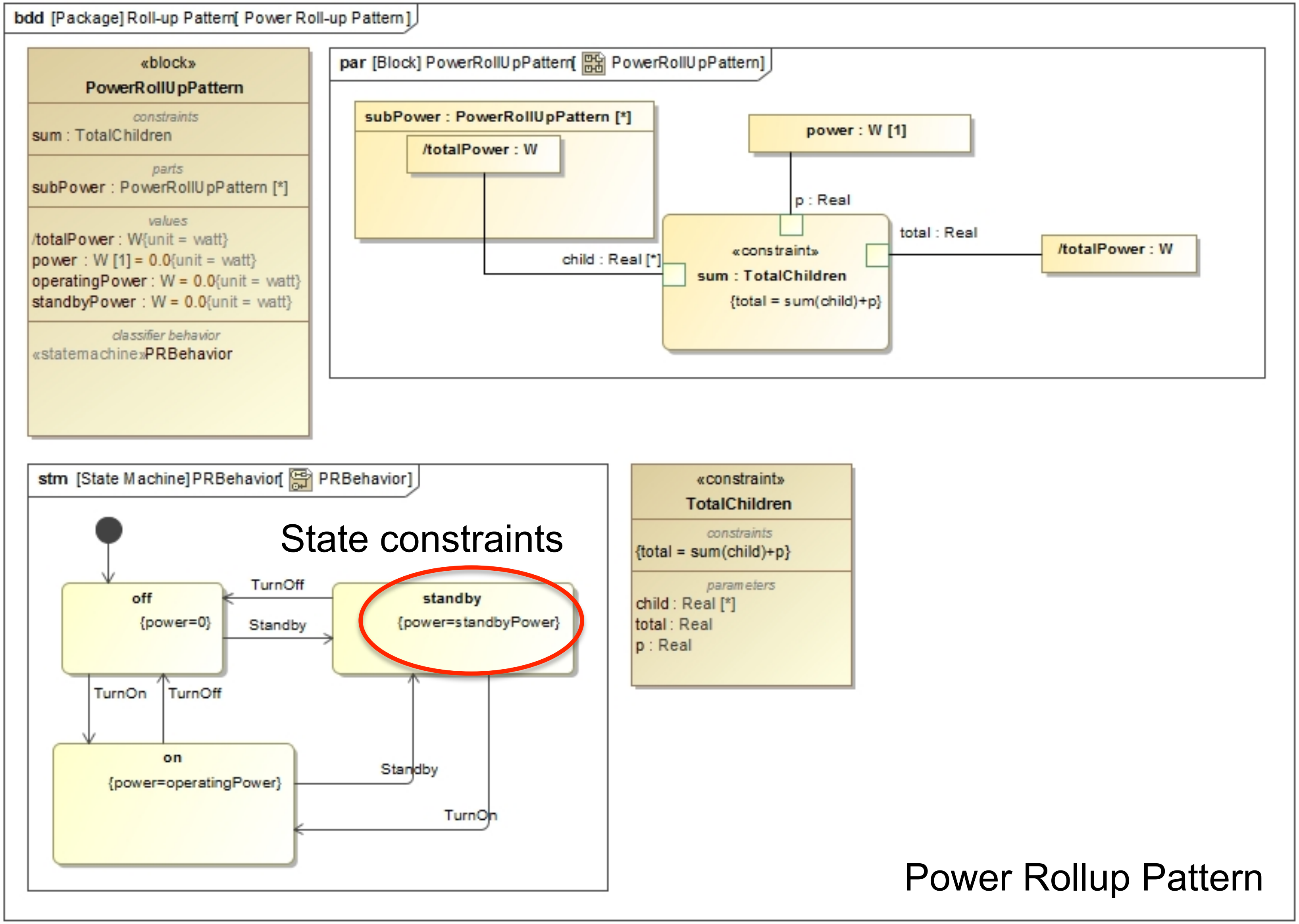
Step 3: Characterize Components

- Add relevant patterns to the design Block to make it executable
- Example: Roll-up Pattern
  - Constrained value represents an aggregate value that is propagating up a hierarchy of subcomponents
  - Static roll-up (e.g., mass roll-up)
  - Dynamic roll-up (e.g., power roll-up)

Parametric Rollup Pattern

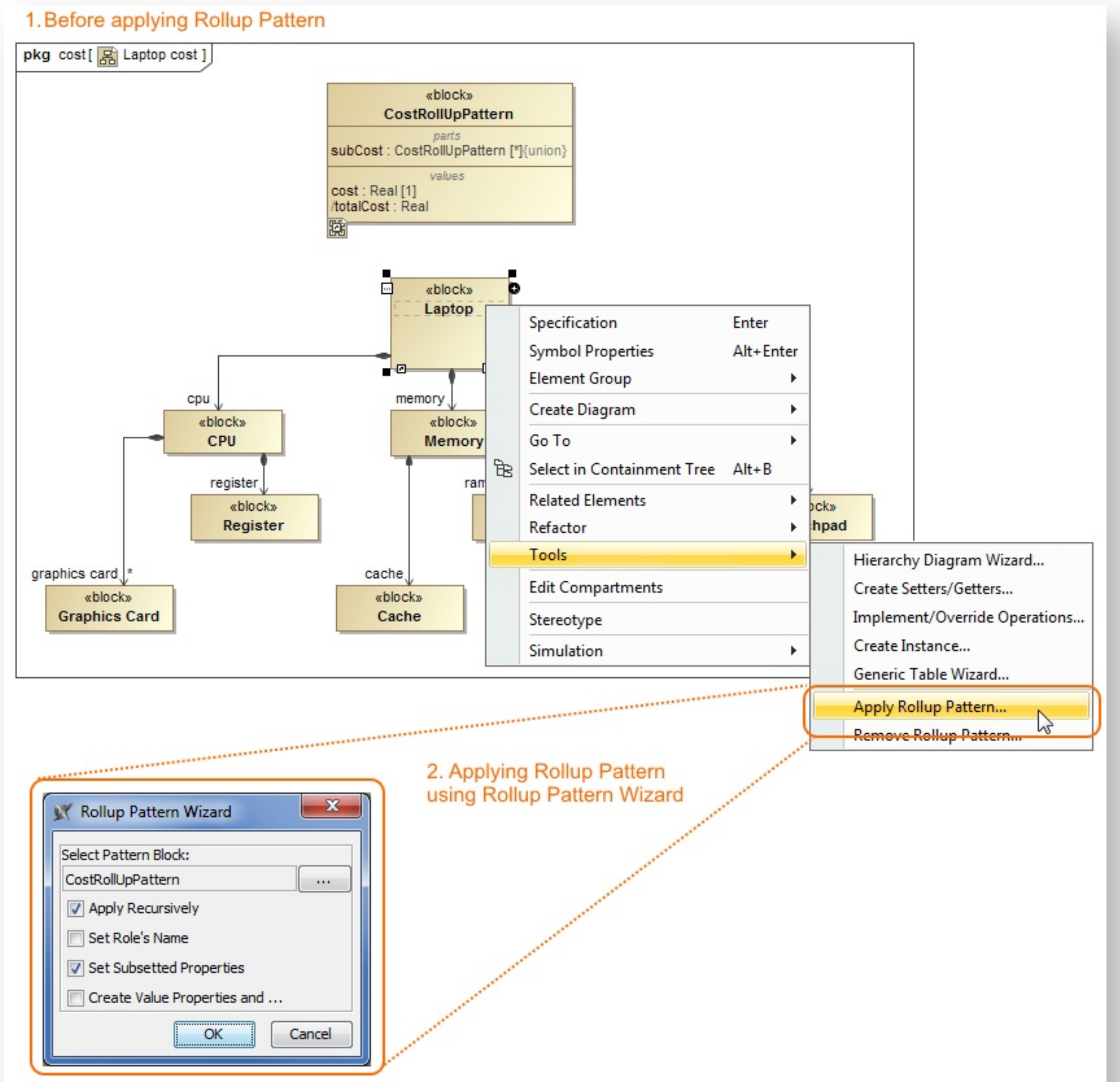


Step 3: Characterize Components

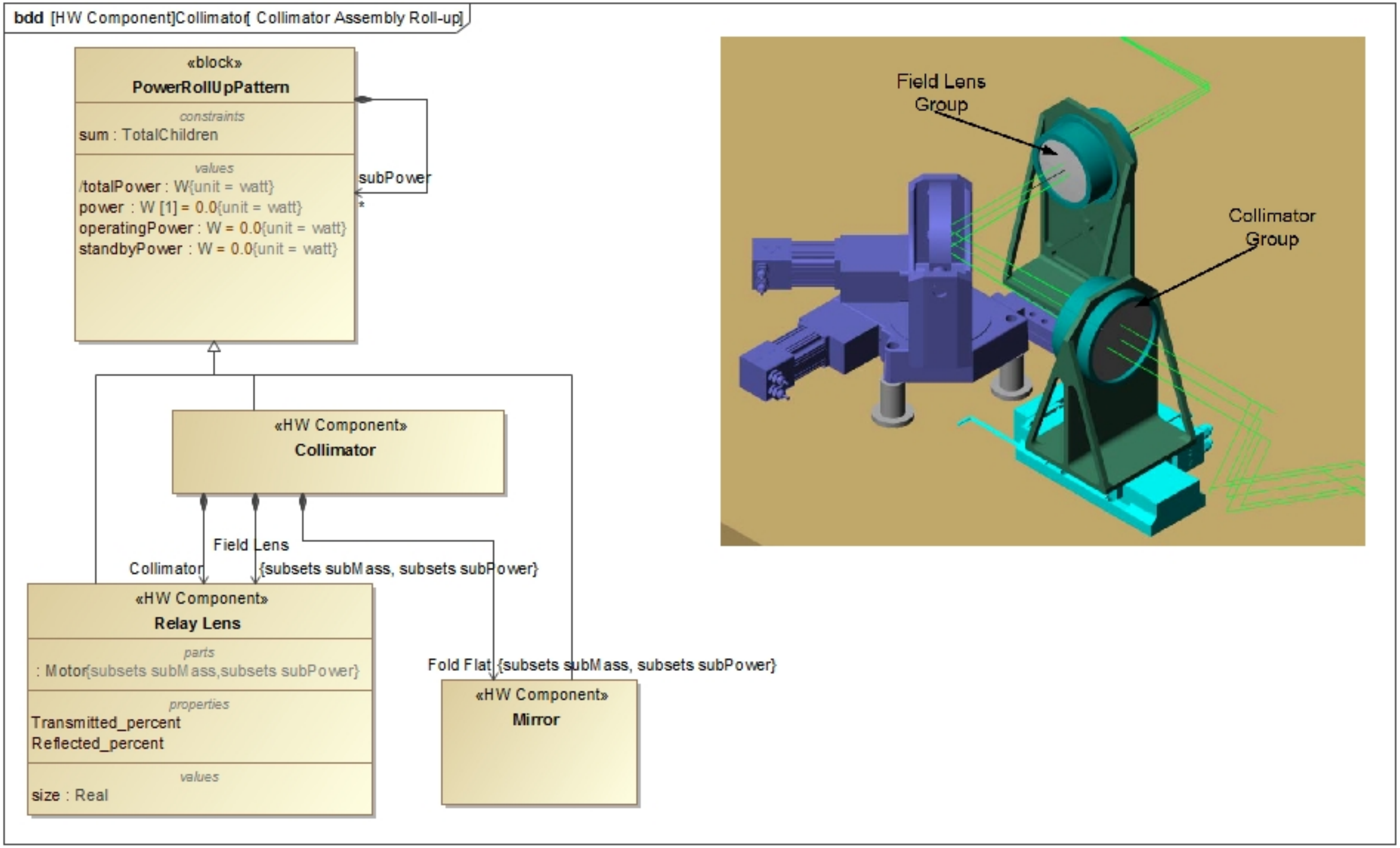


Power Rollup Pattern

Automation


1. Before applying Rollup Pattern
pkg cost[Laptop cost]
«block»
CostRollUpPattern
parts
subCost : CostRollUpPattern [*]{union}
values
cost : Real [1]
/totalCost : Real
«block»
Laptop
cpu
«block»
CPU
memory
«block»
Memory
register
«block»
Register
graphics card
«block»
Graphics Card
cache
«block»
Cache
Specification Enter
Symbol Properties Alt+Enter
Element Group
Create Diagram
Go To
Select in Containment Tree Alt+B
Related Elements
Refactor
Tools
Edit Compartments
Stereotype
Simulation
Hierarchy Diagram Wizard...
Create Setters/Getters...
Implement/Override Operations...
Create Instance...
Generic Table Wizard...
Apply Rollup Pattern...
Remove Rollup Pattern...
2. Applying Rollup Pattern
using Rollup Pattern Wizard
Rollup Pattern Wizard
Select Pattern Block:
CostRollUpPattern
Apply Recursively
Set Role's Name
Set Subsetted Properties
Create Value Properties and ...
OK
Cancel

Step 3: Characterize Components

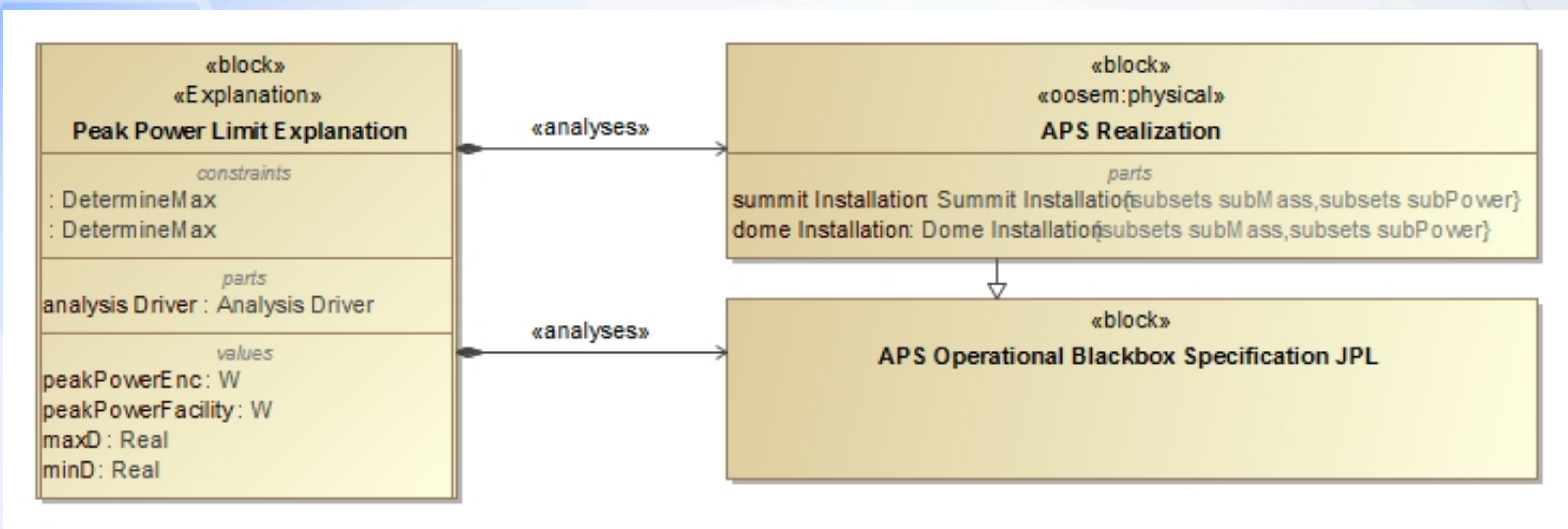


Power Roll-up Pattern Application

Step 4: Specify Analysis Context

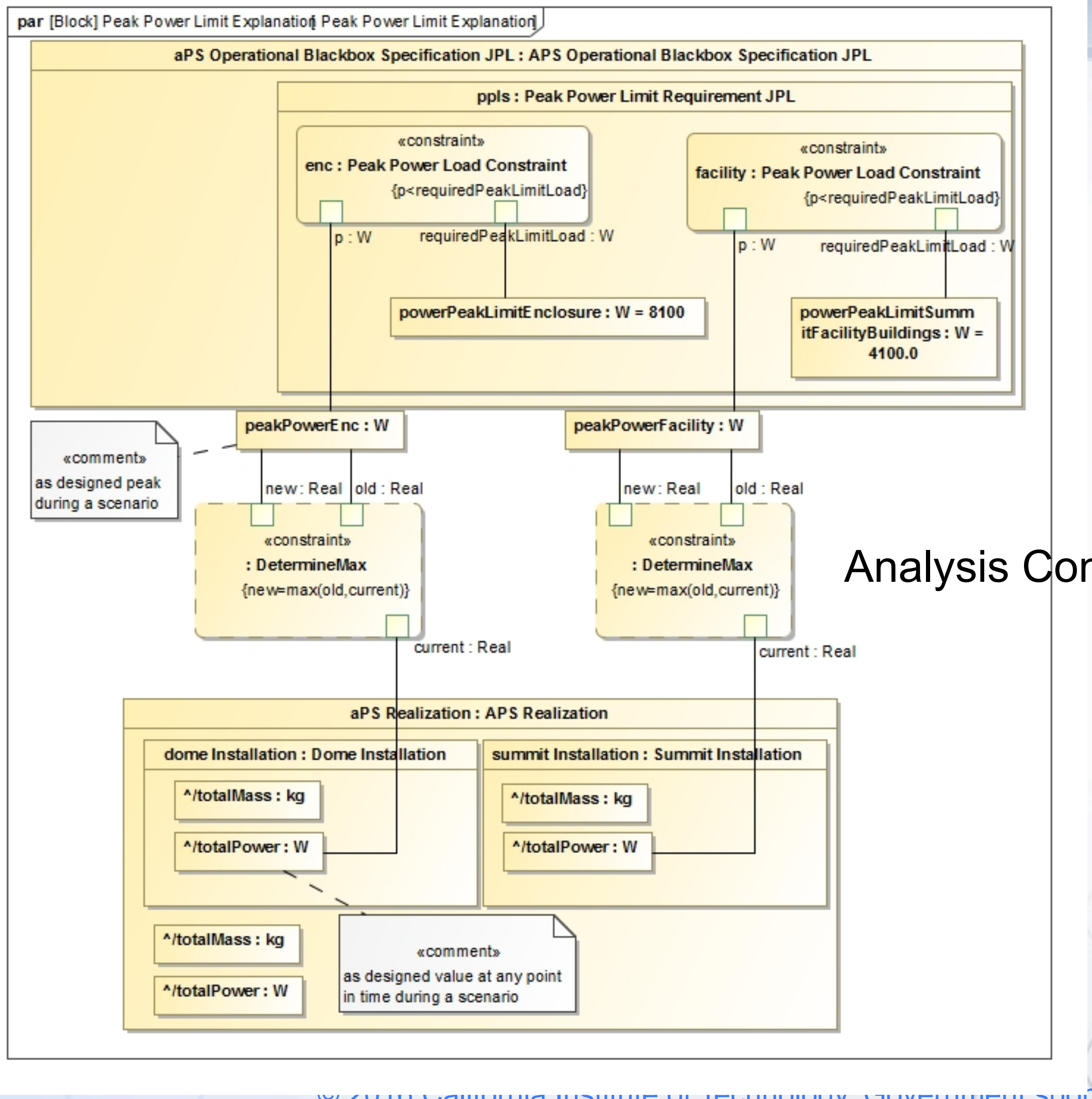
- Analysis Context Pattern
  - Abstract analysis context Block composes both the design black box Block and white box Block
  - Analysis properties defined on the analysis context Block (e.g., peak power, power margin)
  - Analysis parametric model on the analysis context that computes and binds analysis values

Step 4: Specify Analysis Context



Analysis Context Pattern

Step 4: Specify Analysis Context

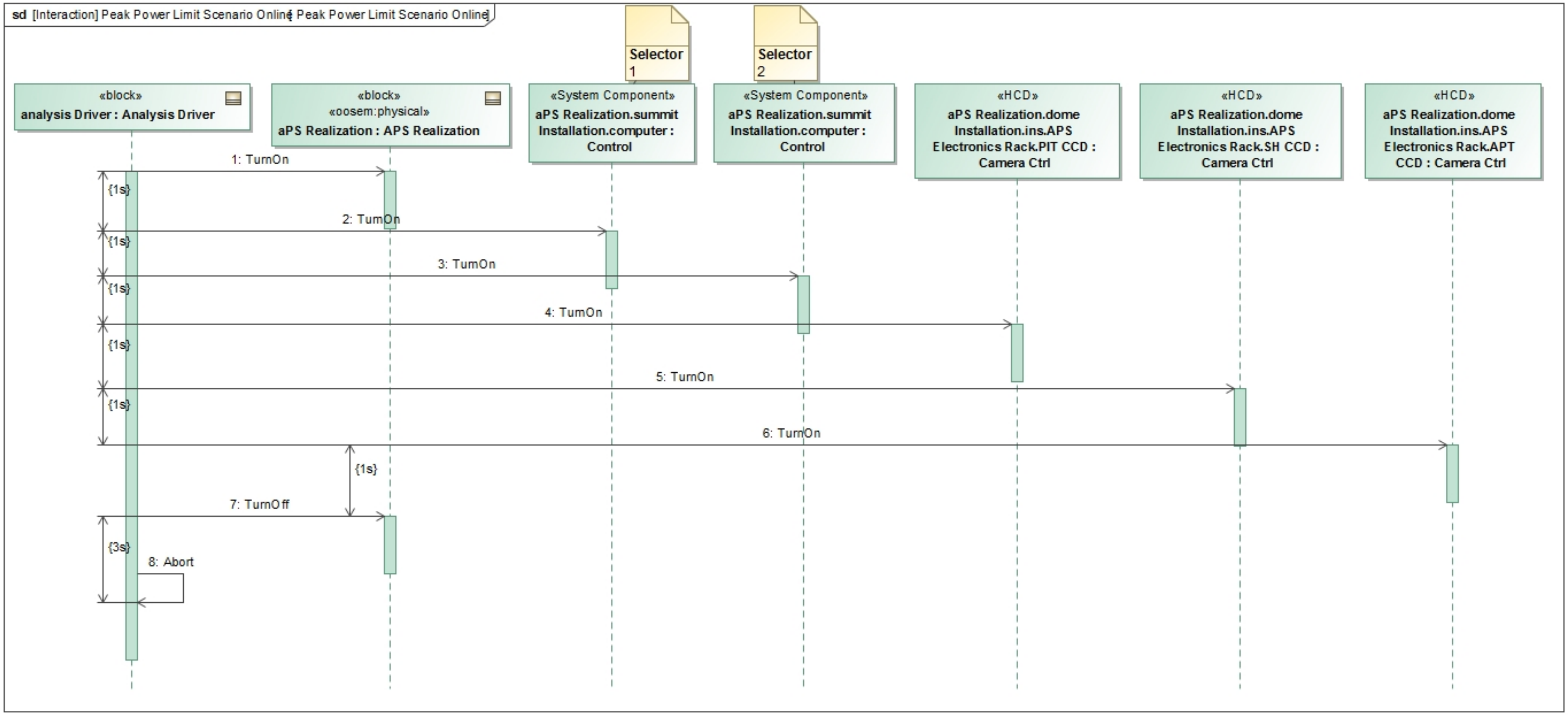


Analysis Context Parametric Model

Step 5: Specify Operational Scenarios

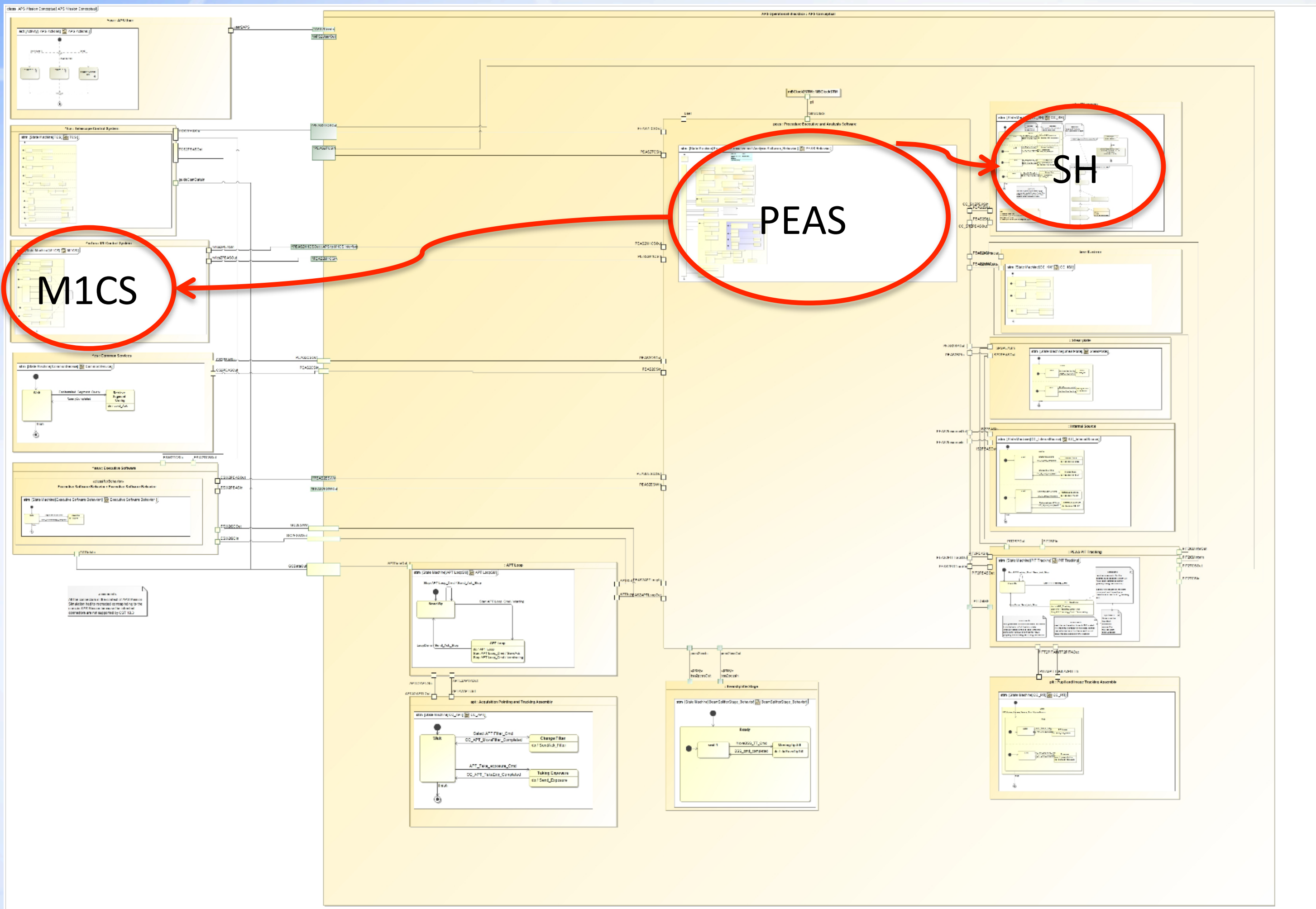
- Operational Scenario Pattern
  - Concrete analysis context Block that
    - Represents one operational scenario (e.g., power configuration)
    - Specializes the abstract analysis context Block
    - Redefines context's properties with scenario-specific values
    - Defines an owned behavior (sequence diagram) as scenario driver
      - Changes the states of the different components, by sending them signals, causing the rolling-up to occur automatically
      - Can specify duration constraints to time the injection of signals thus controlling time spent in a certain state
      - Can use state constraints (on components) to verify during execution if a component is actually in expected state

Step 5: Specify Operational Scenarios



Operational Scenario Driver

Step 2: Conceptual Model



Communication between state machine specified components over ports

nowledged.

Step 6: Specify Scenario Configurations

- Scenario Condition Pattern
  - A decomposition tree of instance specifications representing the state of the scenario
    - Can be presented in tabular form
      - Rows represent the instance specifications (e.g., component)
      - Columns represent values (e.g., operating power) from the instance specifications
- Issues
  - Hard to keep instance specifications in sync with Block hierarchy
    - Mitigation: tool automation
  - Instance specifications cannot be displayed in IBDs
    - Mitigation: use full specialization tree of singleton Blocks for each scenario

Step 6: Specify Analysis Configurations

#	Name	Classifier	Operating Power : W	Standby Power : W
1	peak Power Limit Scenario Online.aPS Realization	APS Realization	0.0	0.0
2	peak Power Limit Scenario Online.aPS Realization.dome Installation	Dome Installation	0.0	0.0
3	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins	Instrument	0.0	0.0
4	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack	Controller Rack	0.0	0.0
5	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt bs	Motor Ctrl	0.0	0.0
6	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt bs.subMass[1]	MassRollUpPattern		
7	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt bs.subPower[1]	PowerRollUpPattern		
8	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt ccd	Camera Ctrl	150.0	200.0
9	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt ccd.subMass[1]	MassRollUpPattern		
10	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt ccd.subPower[1]	PowerRollUpPattern		
11	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt filter 1	Slide Wheel Ctrl	0.0	0.0
12	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt filter 1.subMass[1]	MassRollUpPattern		
13	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt filter 1.subPower[1]	PowerRollUpPattern		
14	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt filter 2	Slide Wheel Ctrl		
15	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.apt filter 3	Slide Wheel Ctrl		
16	peak Power Limit Scenario Online.aPS Realization.dome Installation.ins.aps electronics rack.pit ccd	Camera Ctrl	150.0	100.0

Scenario Initial Condition Pattern

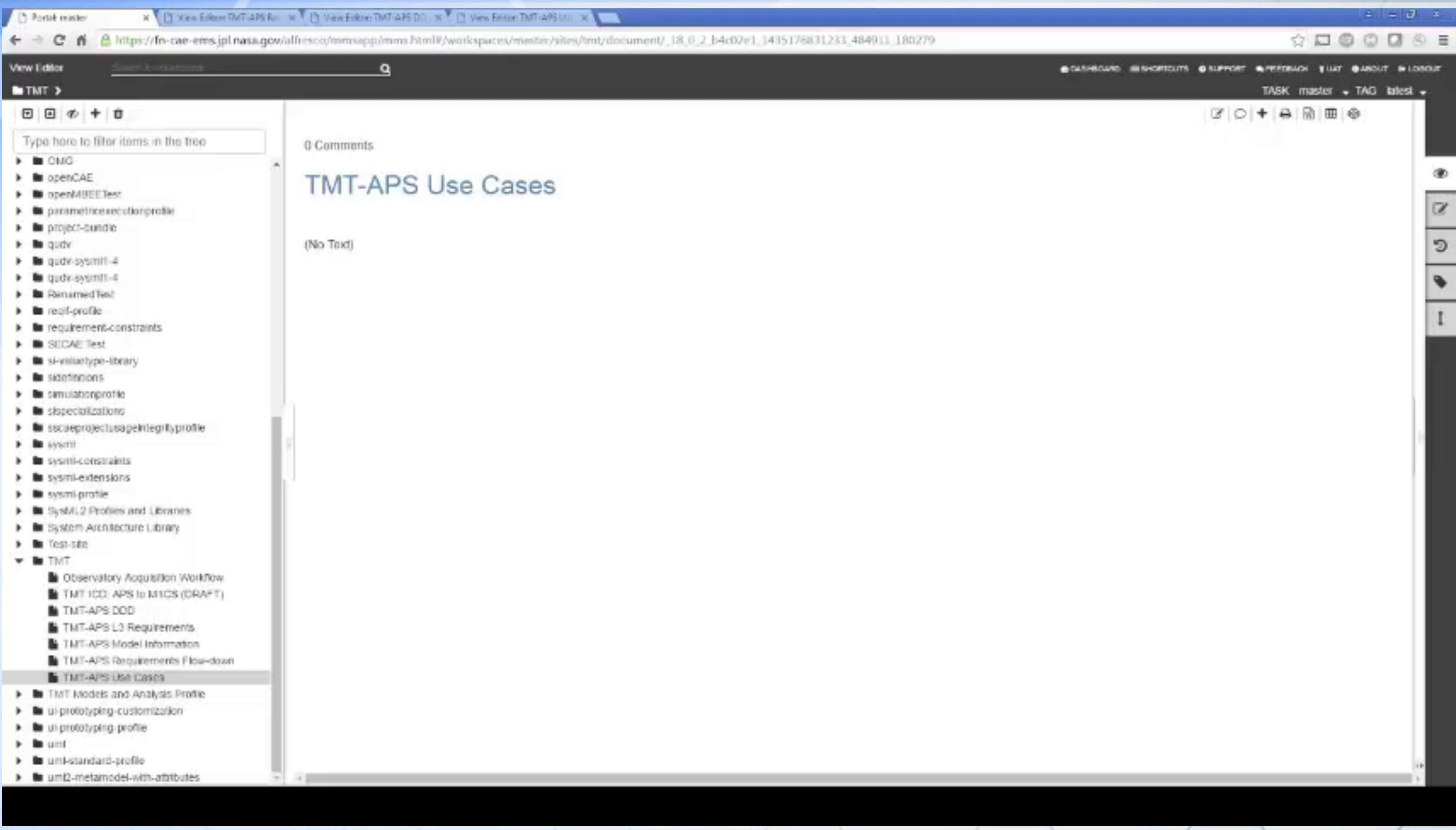
178	peak Power Limit Scenario Online.aPS Realization.summit Installation.subMass[1]	MassRollUpPattern				
179	peak Power Limit Scenario Online.aPS Realization.summit Installation.subPower[1]	PowerRollUpPattern				
180	peak Power Limit Scenario Online.aPS Operational Blackbox Specification JPL.pplc	Peak Power Limit Requirem				8500.0
181	peak Power Limit Scenario Online.aPS Operational Blackbox Specification JPL.ppls	Peak Power Limit Requirem				8100.0
182	peak Power Limit Scenario Online.aPS Realization.pplc	Peak Power Limit Requirem				
183	peak Power Limit Scenario Online.aPS Realization.ppls	Peak Power Limit Requirem				
184	peak Power Limit Scenario Online.aPS Realization.summit Installation.computer1	Control	500.0	100.0	0.0	

Step 7: Run Analysis

- Run the configured analysis with a simulation engine on the initial conditions to get the final conditions:
- 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

TMT
Thirty Meter Telescope

System Level Analysis



TMT
Thirty Meter Telescope

TMT Model

- Customer and supplier model in same SysML project, APS and Adaptive Optics
- Project level (customer) conceptual elements re-used for simulation in downstream design
- Analysis: Duration, Power and Mass

https://github.com/Open-MBEE/TMT-SysML-Model

Summary & Outlook

- It is possible to automate requirements verification in SysML models
- Introduced a new Executable System Engineering Method that consists of a set of pure SysML analysis patterns
- The method can be executed using an Off the shelf simulation engine for SysML
- big interest in other projects at JPL

- Trigger Analysis from Web interface and auto-generate documents
- Integrate analysis engine (solver)
- Integrate FMI units

TMT
Thirty Meter Telescope

Thank You !

Visit No Magic booth at H1-H2

TMT
Thirty Meter Telescope

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and No Magic Inc.

The TMT Project gratefully acknowledges the support of the TMT collaborating institutions. They are the Association of Canadian Universities for Research in Astronomy (ACURA), the California Institute of Technology, the University of California, the National Astronomical Observatory of Japan, the National Astronomical Observatories of China and their consortium partners, and the Department of Science and Technology of India and their supported institutes. This work was supported as well by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the National Research Council of Canada, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Universities for Research in Astronomy (AURA) and the U.S. National Science Foundation.

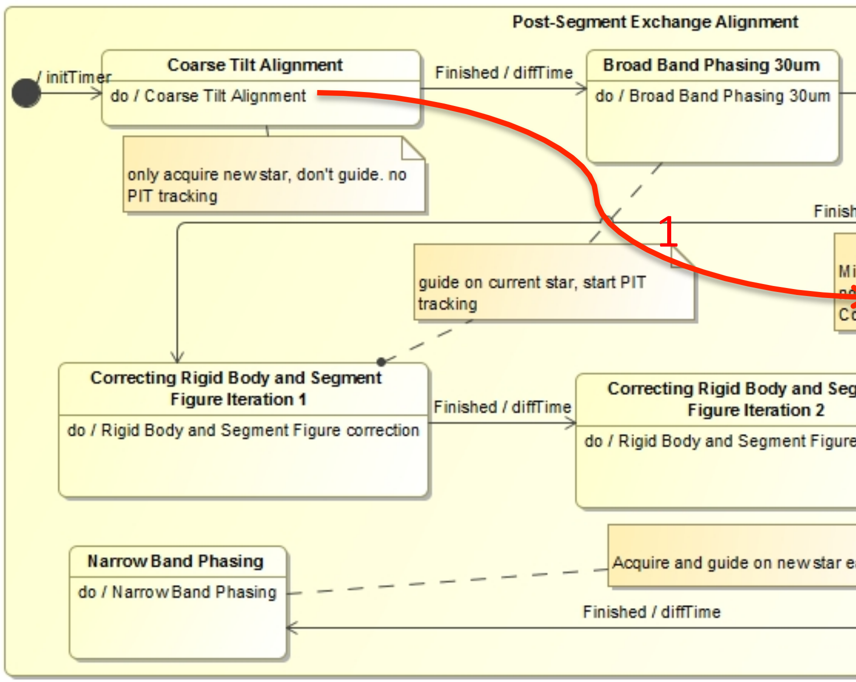
Reference

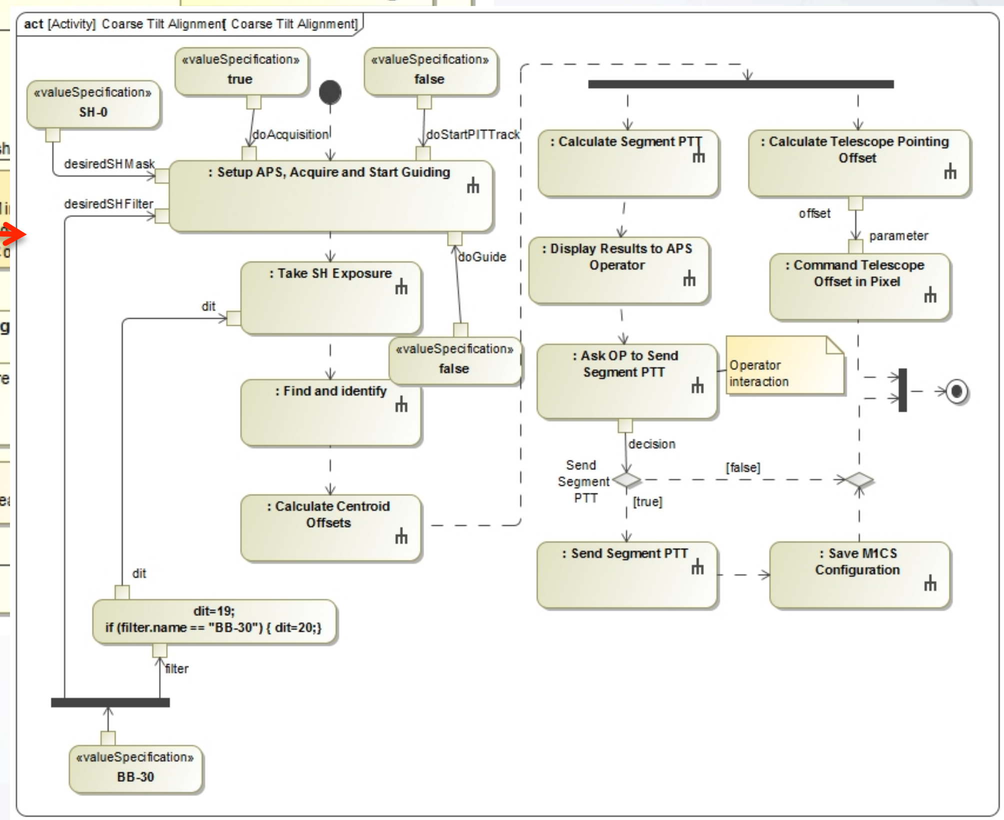
- Karban, R., Jankevičius, N., Elaasar, M. “ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns”, (to appear in the proceedings of INCOSE International Symposium (IS), Edinburgh, Scotland, 2016.)
- Karban, R., “Using Executable SysML Models to Generate Systems Engineering Products”, NoMagic World Symposium, Allen, TX, 2016
- Open Source TMT model: https://github.com/Open-MBEE/TMT-SysML-Model
- Open Source Engineering Environment: https://github.com/Open-MBEE
- A Practical Guide to SysML, 3rd Edition, Chapter 17 by Friedenthal, Moore, and Steiner
- Zwemer, D., “Connecting SysML with PLM/ALM, CAD, Simulation, Requirements, and Project Management Tools”, May 2016

Backup

TMT
Thirty Meter Telescope

Conceptual Design



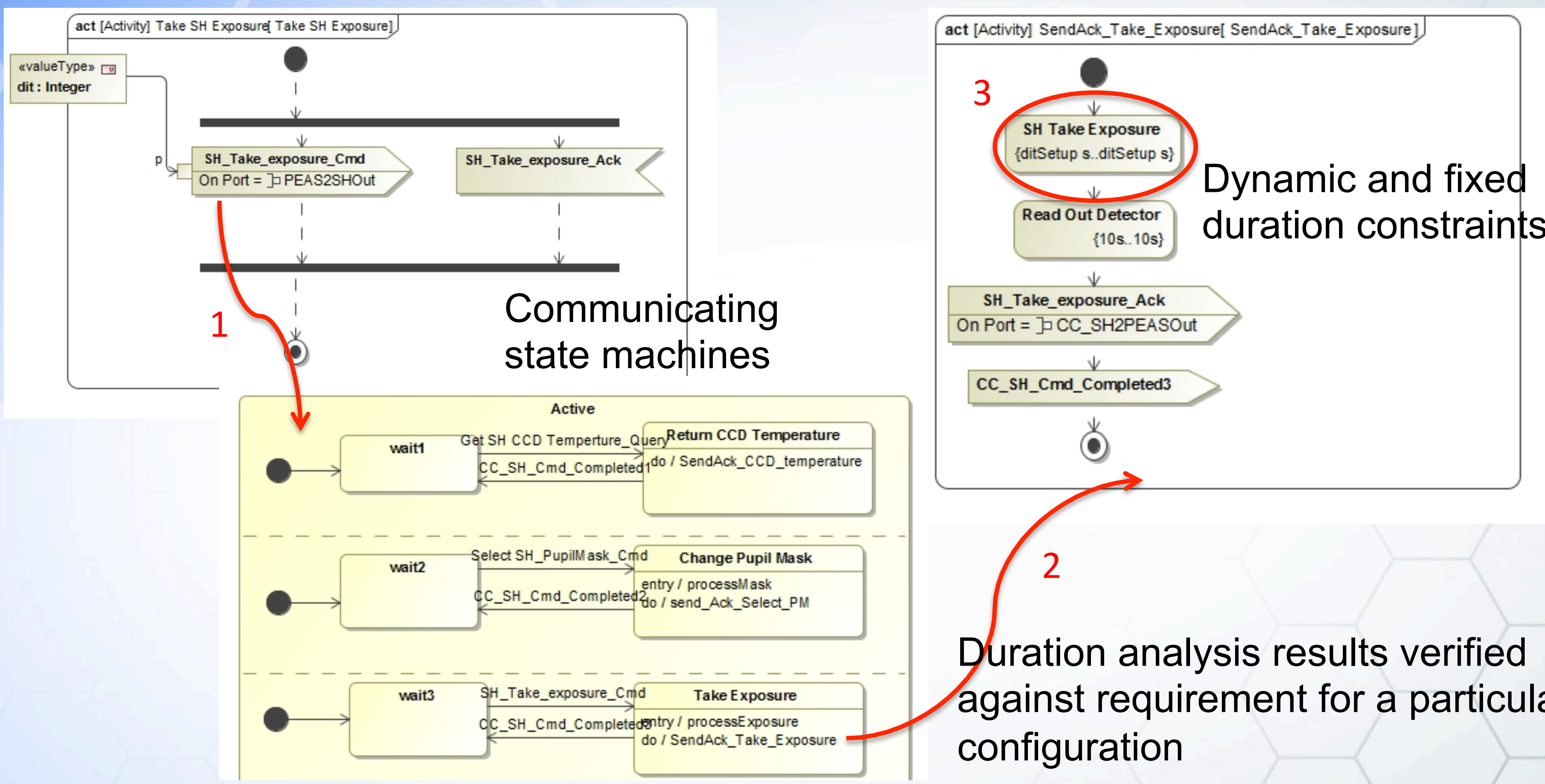


Operational behavior captured with state machines and activity models

2

#	Name	Specification	Constrained Element	Owner Of Constrained Element
1	dAPTMoving	10s..10s	Moving	SendAck_Filter
2	dAPTTakeExposure	ditSetup s..ditSetup s	APT Take Exposure	Send_Exposure
3	dCalcCentroidOffsets	0s..1s	Execute Centroid Offset Calculation	Calculate Centroid Offsets
4	dCalcPupilRegistrationOffset	0.5s..1s	Calculate Pupil Registration Offset	Calculate Pupil Registration and Image Offset
5	dCalcRMSForM2AndSegmentPTT	1s..3s	Calculate RMS for M2 and Segment PTT Cmds	Calculate RMS for M2 and Segment PTT cmds

Conceptual behavior model



Communicating state machines

Dynamic and fixed duration constraints

Duration analysis results verified against requirement for a particular configuration

4

#	Name	T Final : Real	Post Seg Xchg Time Limit : Second	Post Segment Exchange : Post Segment Exchange Time Constraint	Off Axis Measurement Steps : Integer	Off Axis Map Points : Integer	RB Dit : Integer	Phasing Dit : Integer
1	maintenance Alignment Duration Scenario.aps mission conceptual.aps co		7200.0	pass				
2	maintenance Alignment Duration Scenario.aps mission conceptual.aps co	1517.0			6	7	45	20

OpenMBEE Core Integration