DARPA

National Aeronautics and Space Administration

NASA

Model-based Design and Validation of Cellularized Satellites for the DARPA Phoenix Program

Mitch Ingham, Alek Kerzhner, Omair Khan, Molly Bittner
Jet Propulsion Laboratory, California Institute of Technology

Jaime Ramirez, Javier de Luis, Jeremy Hollman, Justin McClellan
Aurora Flight Sciences

AIAA Infotech@Aerospace Conference
August 19th, 2013

Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

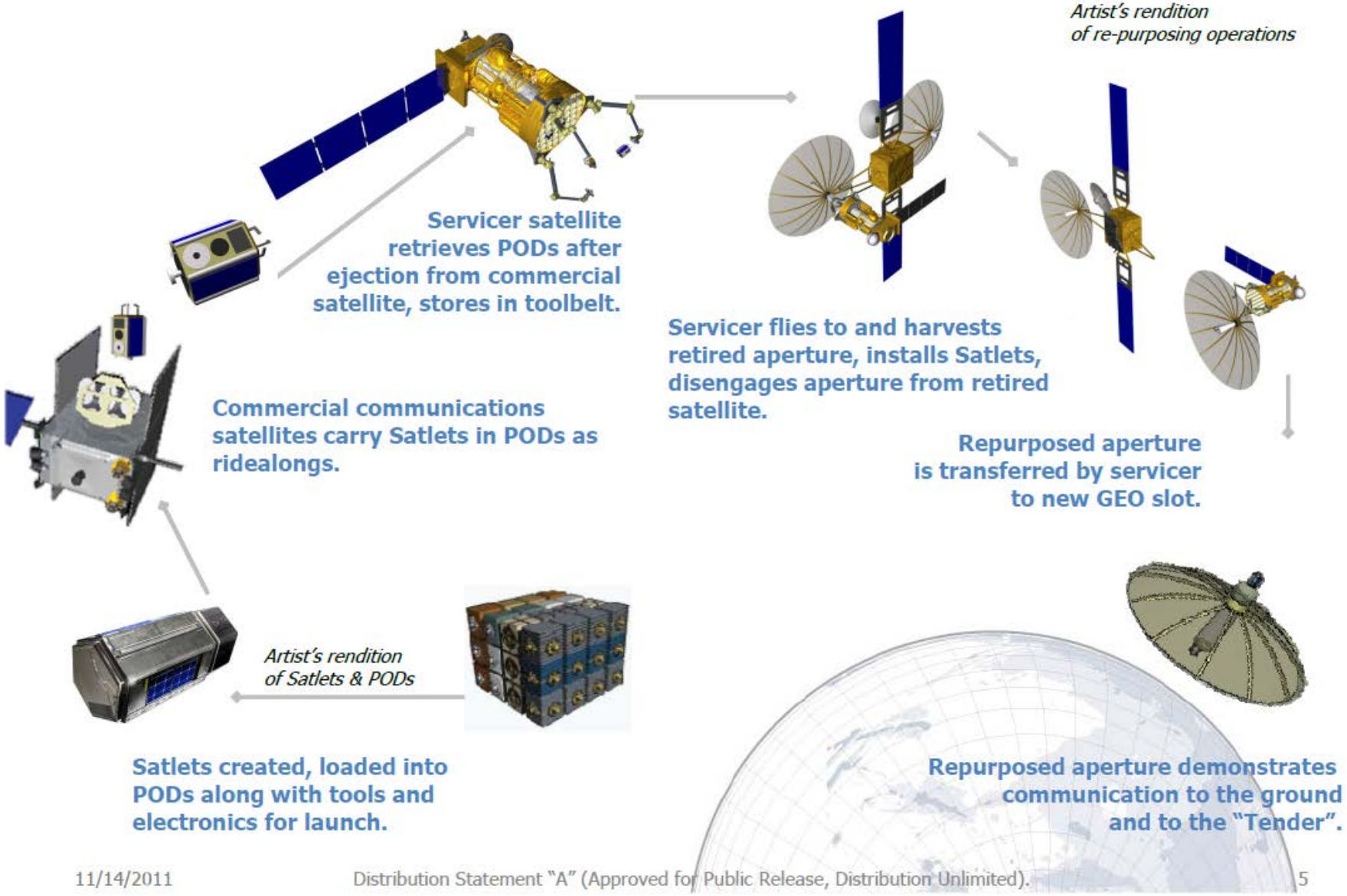
DARPA Phoenix Program

- **The goal of the Phoenix program is to develop and demonstrate technologies to cooperatively harvest and re-use valuable components from *retired, nonworking* satellites in GEO and <u>demonstrate the ability to create new space systems at greatly reduced cost</u>.**

- **Phoenix seeks to demonstrate around-the-clock, globally persistent communication capability for warfighters more economically, by *robotically removing and re-using* GEO-based space apertures and antennas from de-commissioned satellites in the graveyard or disposal orbit.**

- **The Phoenix program is developing a new class of very small 'satlets,' which can be sent to the GEO region more economically as a "ride along" on a commercial satellite launch, and then robotically attached to the antenna of a non-functional cooperating satellite, essentially creating a new space system.**

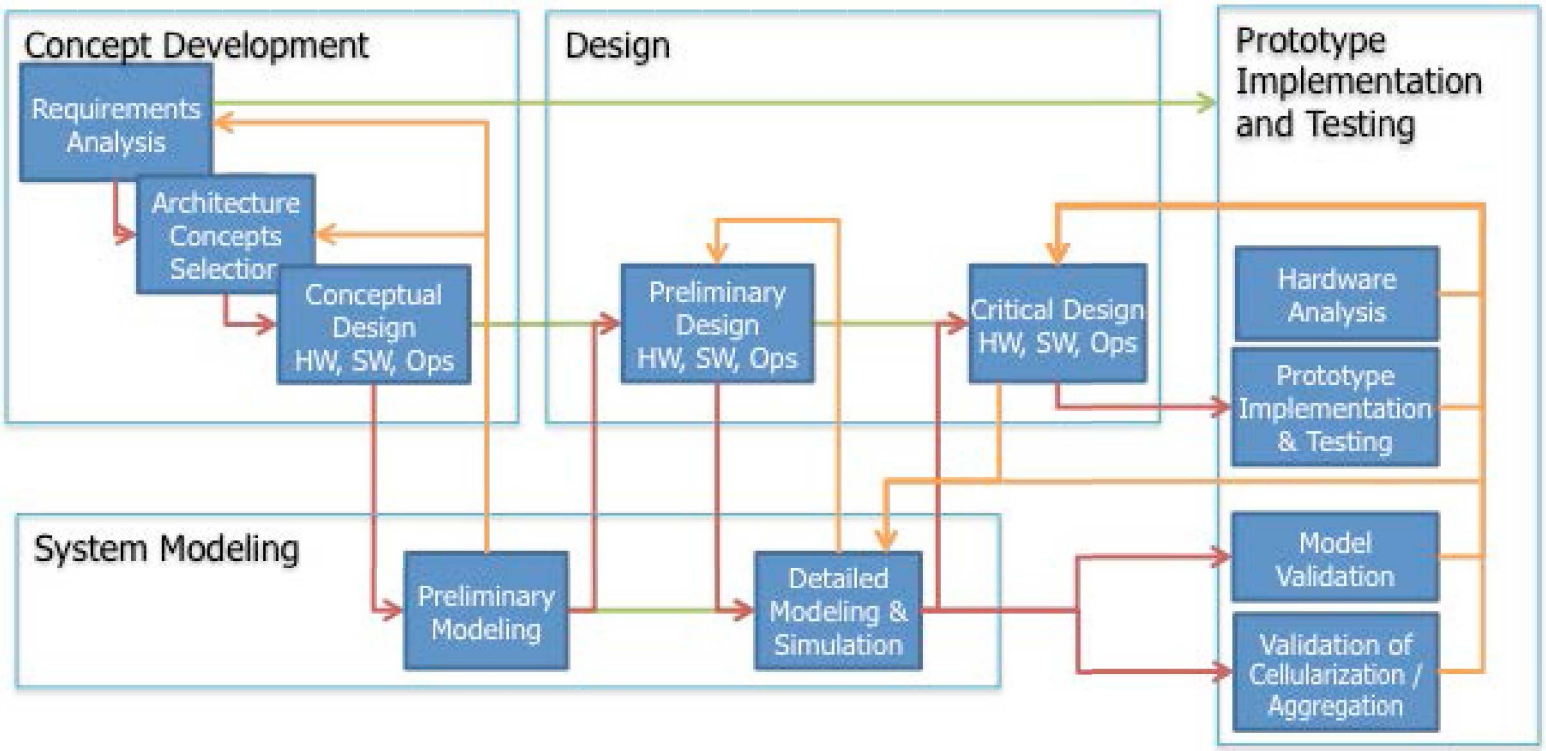
The Phoenix Demonstration Mission



Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

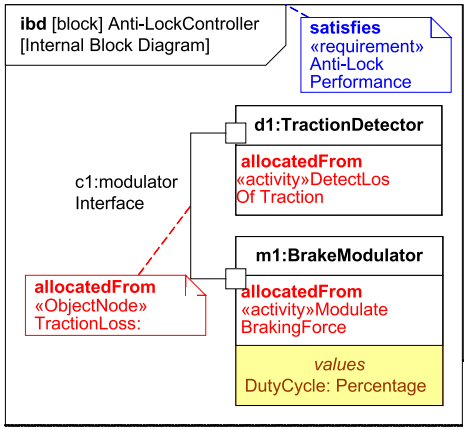
Our Development Process



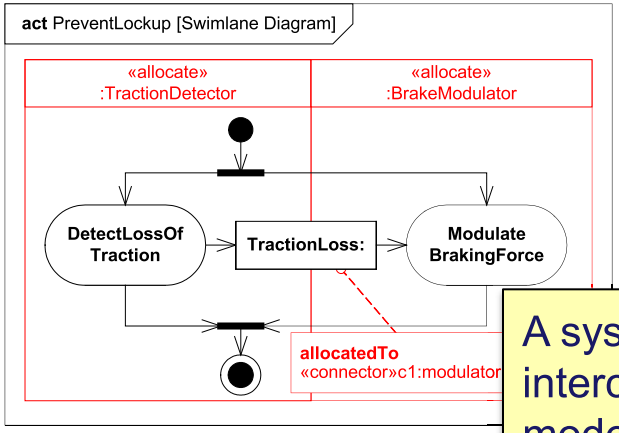
- Concept development phase started from a set of basic requirements and provided guidance for the design.
- Underlying system model provides coherence and captures the design.
- Feedback loops between system modeling task and design tasks provide consistent development.
- Prototype implementation and testing feed from/to design and model.

Background: What is a System Model?

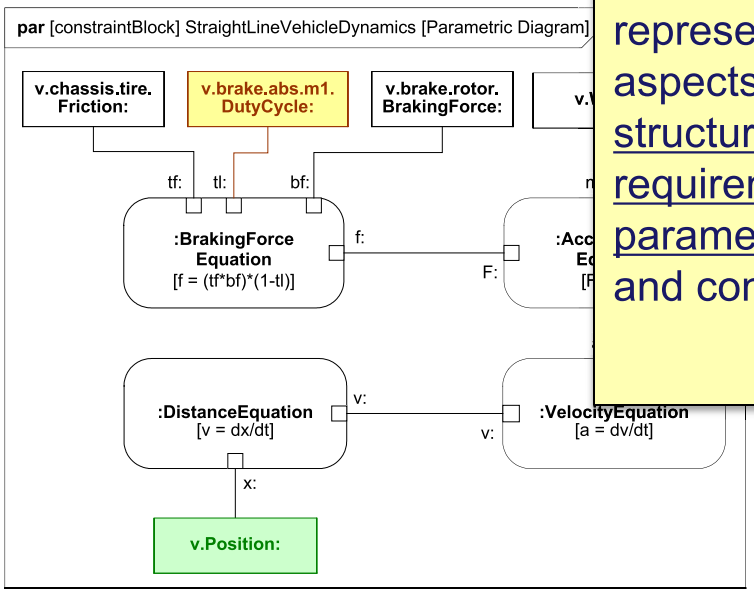
1. Structure

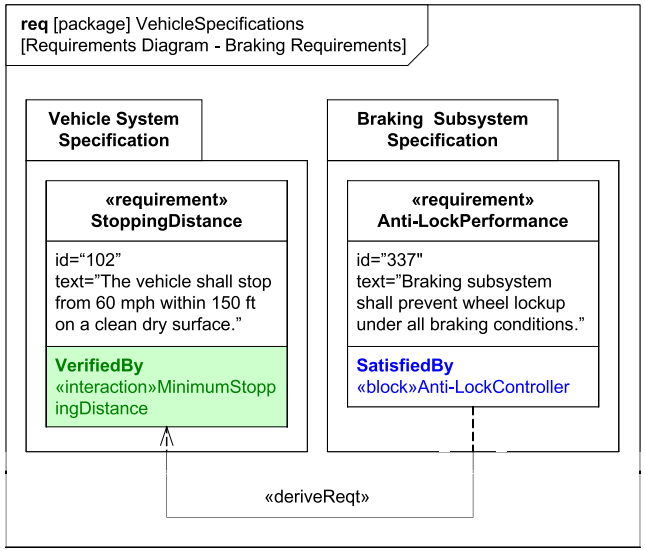


2. Behavior



A system model is an interconnected set of model elements that represent key system aspects including structure, behavior, requirements, and parametrics (equations and constraints).





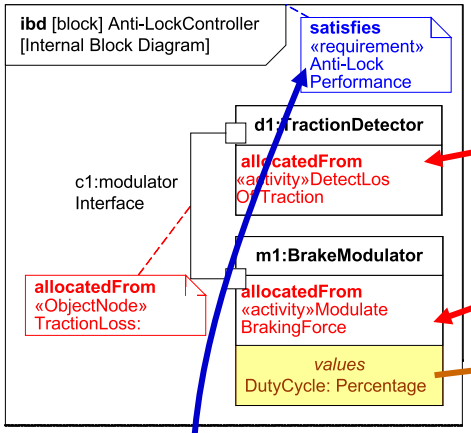
4. Requirements

3. Parametrics

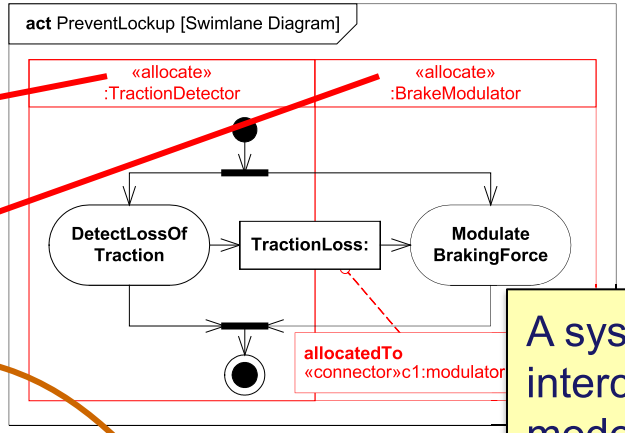
Courtesy S. Friedenthal

Background: What is a System Model?

1. Structure



2. Behavior

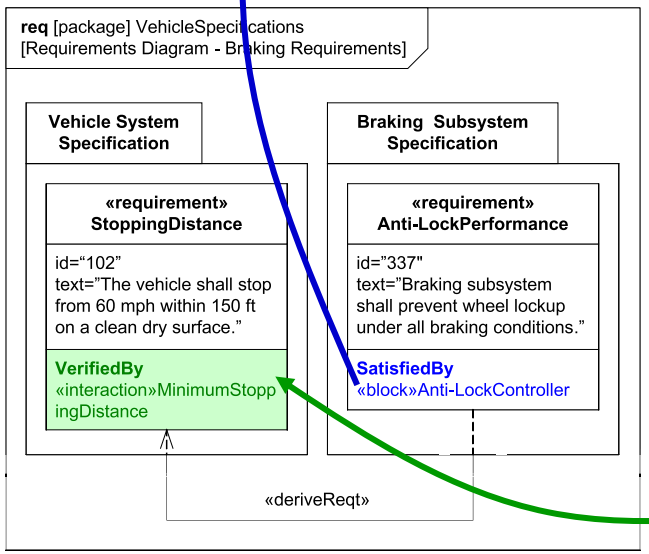


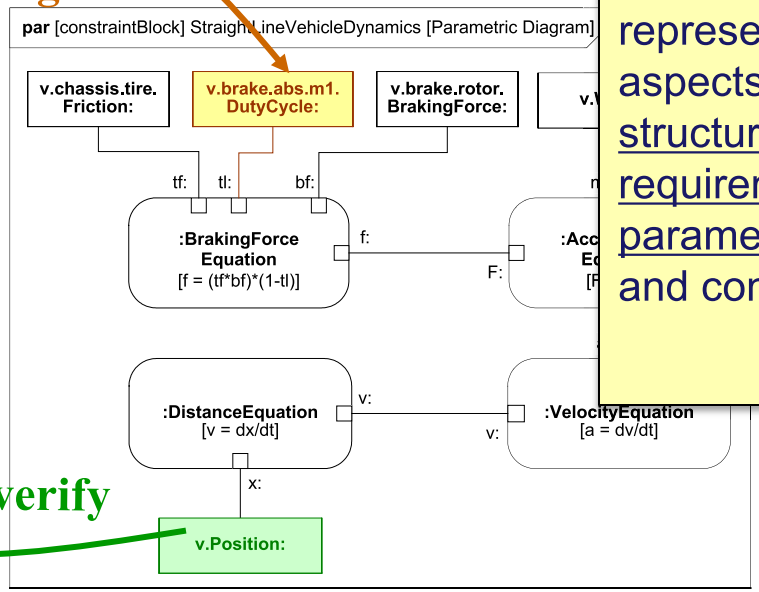
allocate

value binding

satisfy

verify





4. Requirements

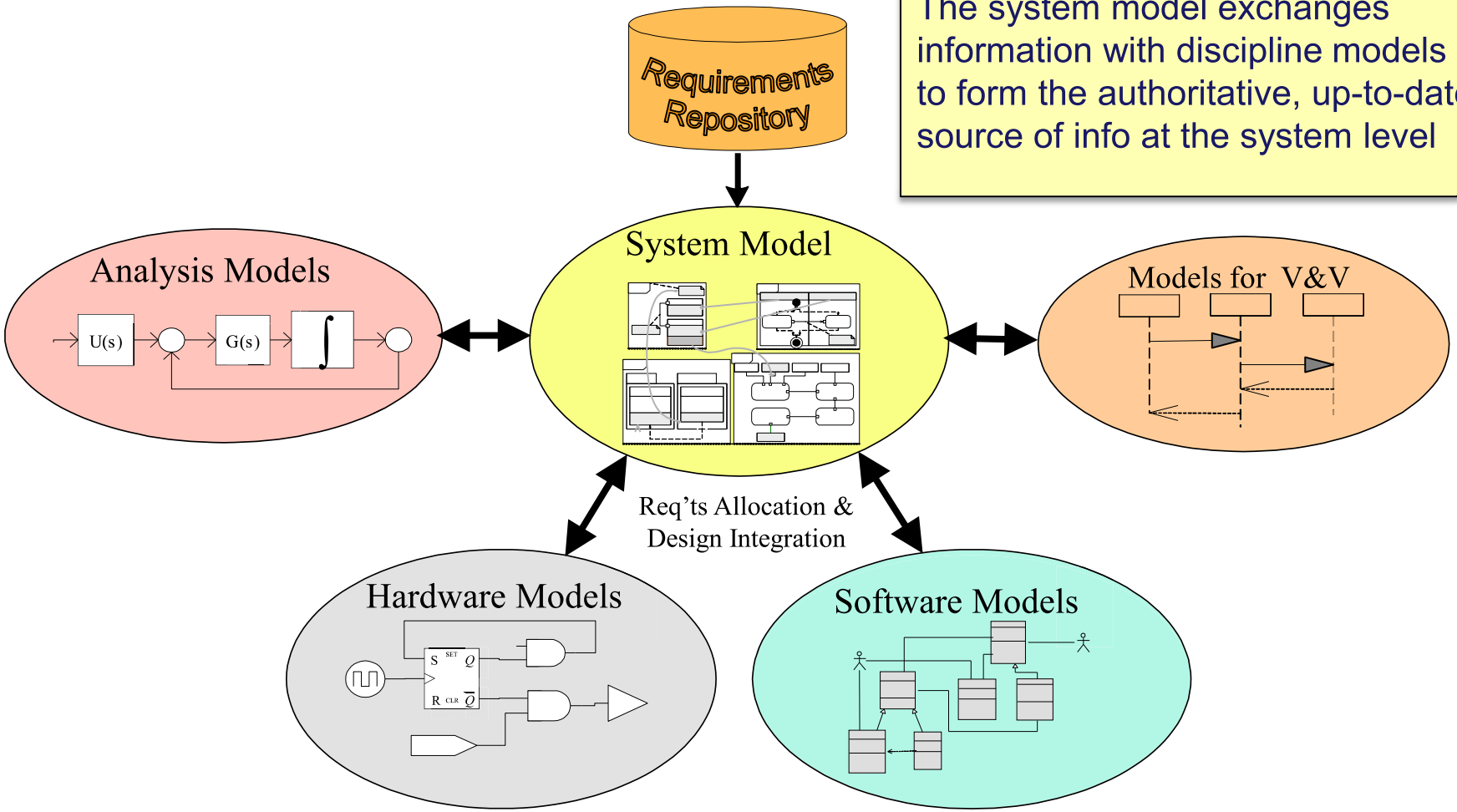
3. Parametrics

A system model is an interconnected set of model elements that represent key system aspects including structure, behavior, requirements, and parametrics (equations and constraints).

Courtesy S. Friedenthal

Integrated Model Centric Engineering: System Model as an Integrating Framework

The system model exchanges information with discipline models to form the authoritative, up-to-date source of info at the system level



Courtesy S. Friedenthal

Integrated Model Centric Engineering: JPL's IMCE Capabilities

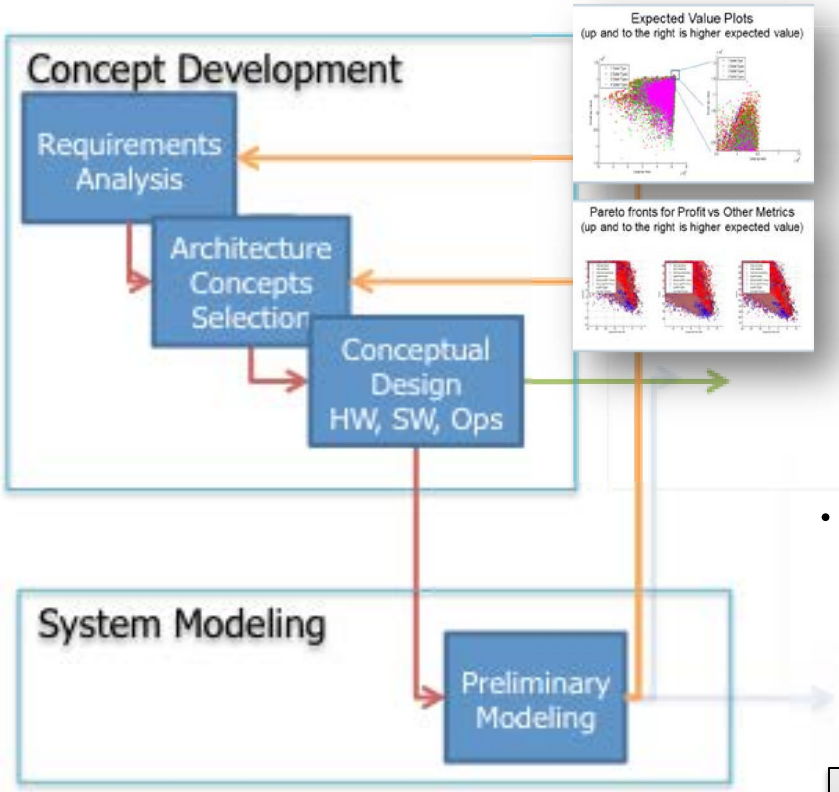
- **Our System Modeling approach is leveraging JPL's Integrated Model-Centric Engineering (IMCE) technologies and capabilities**
  - State-of-the-art system modeling and model-based verification & validation capabilities that have been developed for NASA missions, such as the Europa Habitability Mission, and non-NASA missions including other DARPA programs

- **IMCE capabilities provide a unique system modeling solution for the Phoenix Satlet system**
  - Provides a *standards-based* framework for modeling complex systems that is *analyzable*
  - JPL-developed tools that enable systems engineers to develop system models in SysML, integrate them with discipline-specific models in other languages, perform analyses and prove properties about their models

Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

Model-Based Architecture Selection



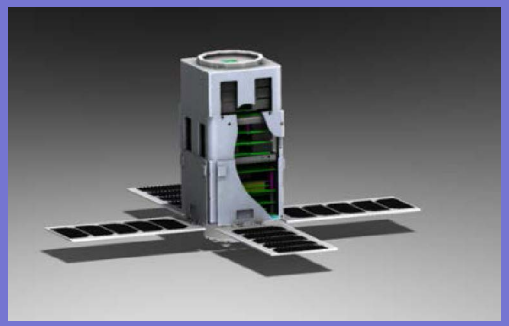
Architecture Trade Analysis Results

1. **Communications & Data handling satlet type** for computation, data processing, centralized attitude sensing and ground communication
2. **Actuation satlet type** for propulsion/attitude actuation
3. **A Payload** to provide specialized mission functionality.
4. **Modules** to provide geometrical/structural behavior.

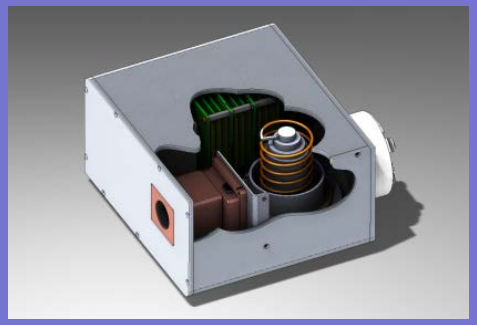
- **Results suggest that a cellularized architecture allows aggregation of capability to efficiently meet requirements for different missions.**

Simplicity at satlet level can still result in required aggregate system behavior, which is the goal of a satlet based system architecture.

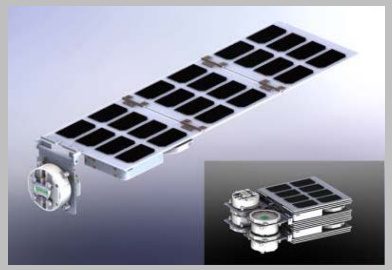
Overview of our Satlet Architecture



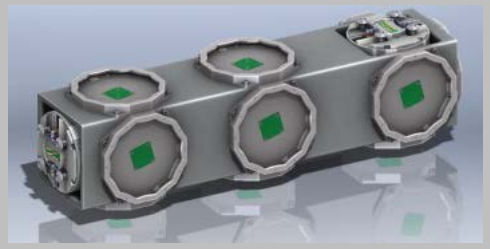
Actuation Satlet : Common avionics box to which actuators can be attached: MEMS micro-spray, Busek, reaction wheels or other actuation modules.



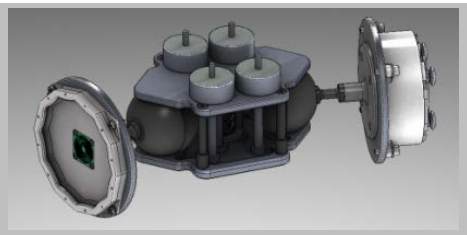
C&DH Satlet: Logical functions rather than motor functions: computation, attitude sensing, power storage and space-to-ground communications that do not need physical/geometrical placement.



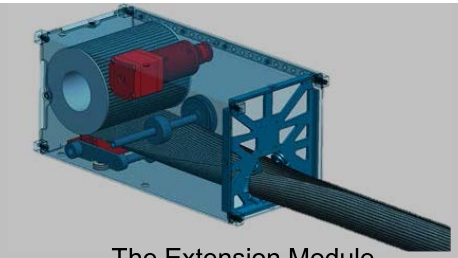
The Solar Collection/Radiator Module



The Truss Module



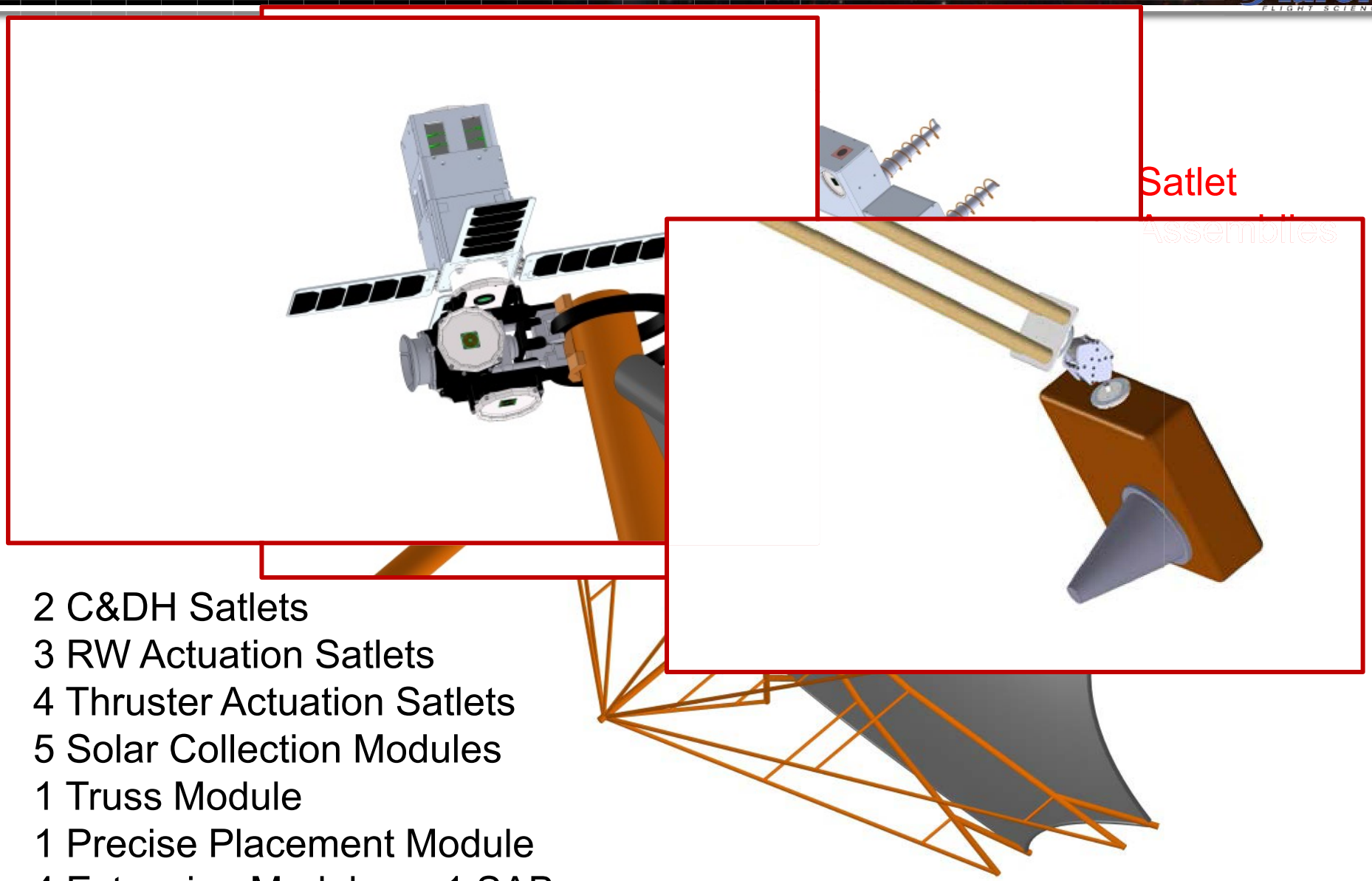
The Precise Placement Module (PPM)



The Extension Module

+ *A Payload* which is specific to the mission

"ApSat": Large Aperture Configuration



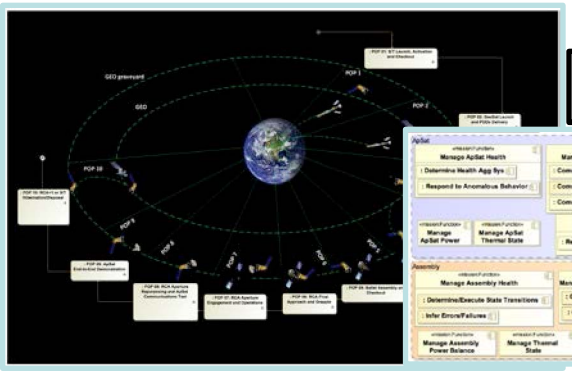
2 C&DH Satlets
3 RW Actuation Satlets
4 Thruster Actuation Satlets
5 Solar Collection Modules
1 Truss Module
1 Precise Placement Module
4 Extension Modules + 1 SAB

Outline

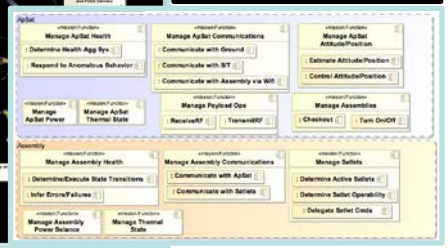
- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

Model-Based Design Approach: Overview

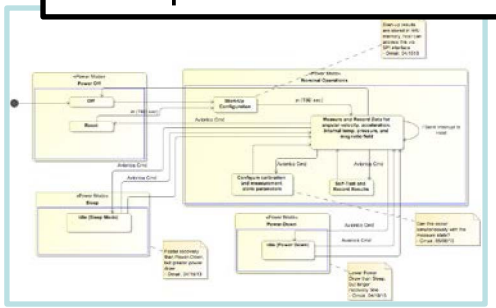
ConOps



Functions

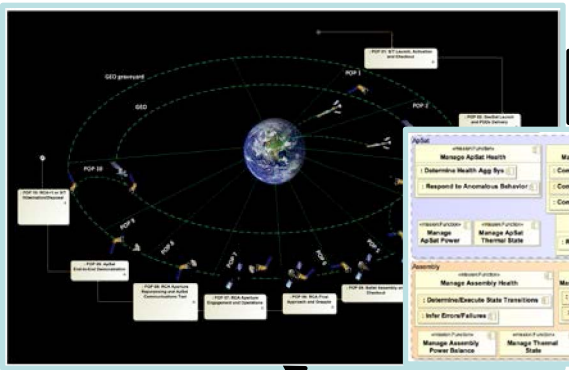


Component Behavior



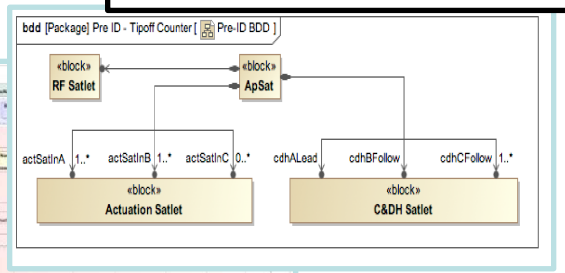
Model-Based Design Approach: Overview

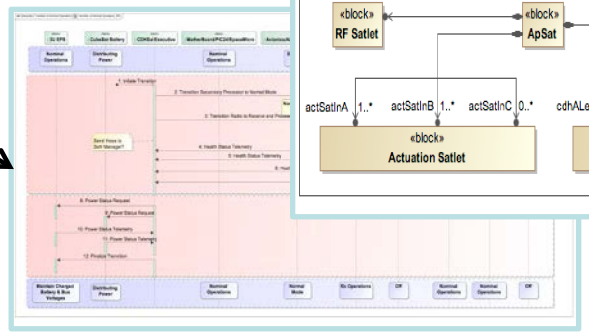
ConOps



Functions

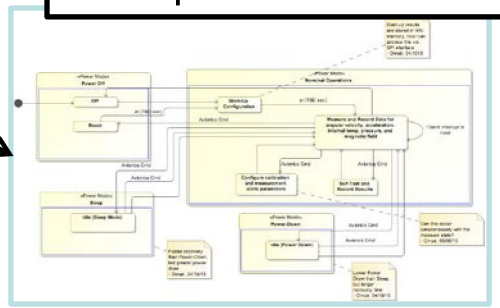
ApSat Configurations





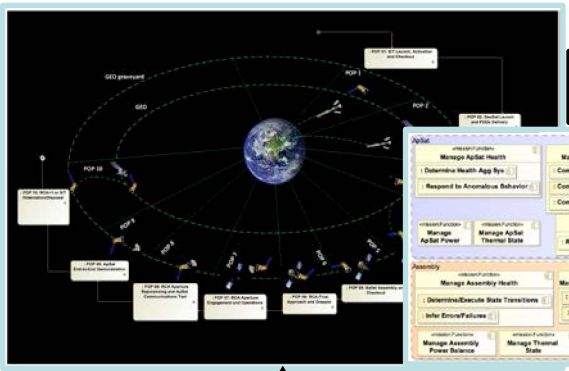
Satlet Structure & Behavior

Component Behavior

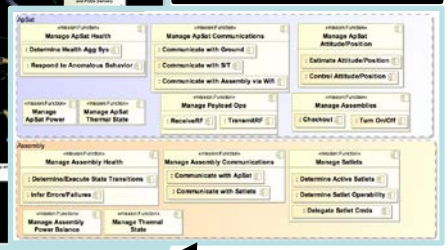


Model-Based Design Approach: Overview

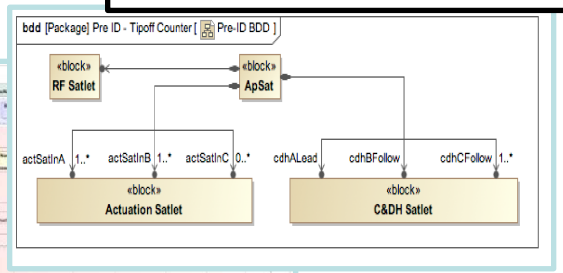
ConOps



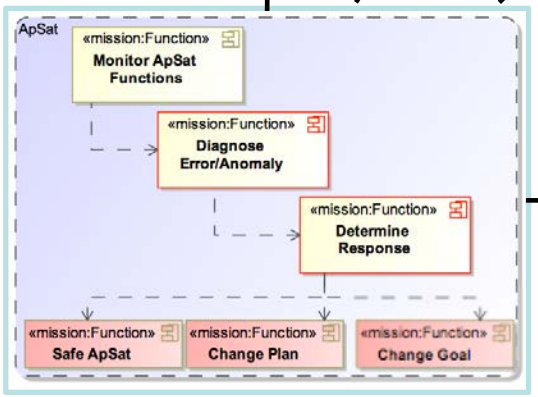
Functions

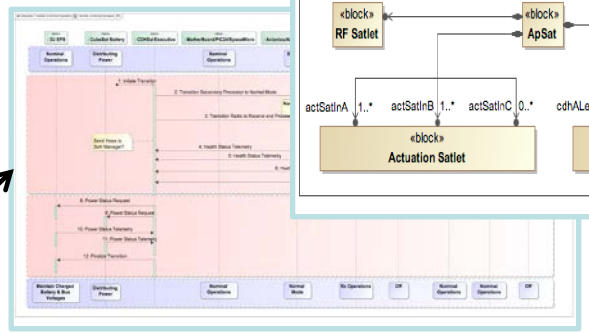


ApSat Configurations



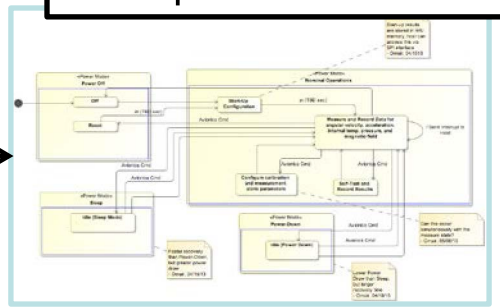
Fault Protection





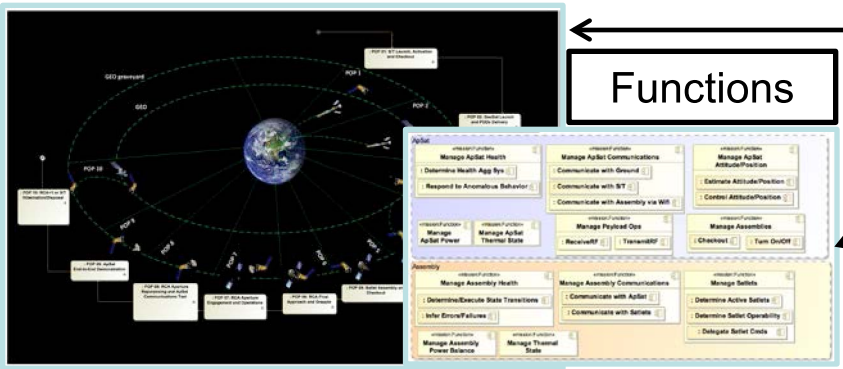
Satlet Structure & Behavior

Component Behavior



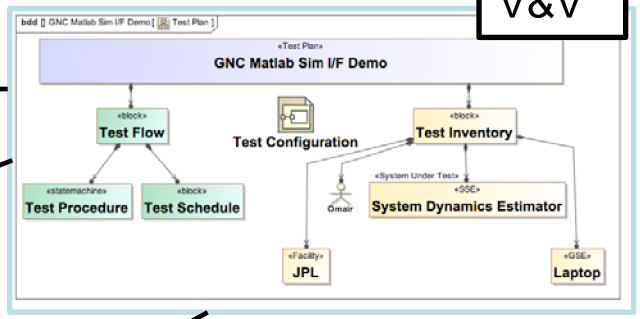
Model-Based Design Approach: Overview

ConOps

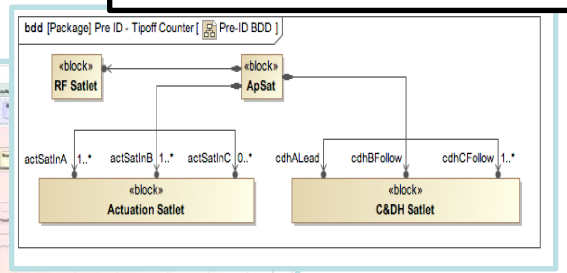


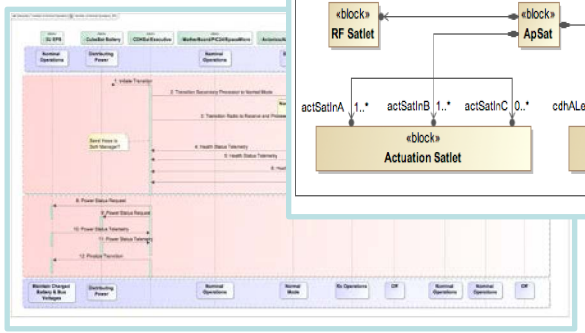
Functions

V&V



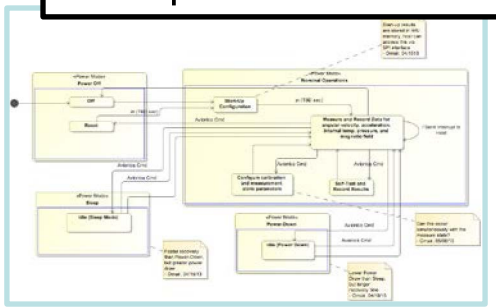
ApSat Configurations



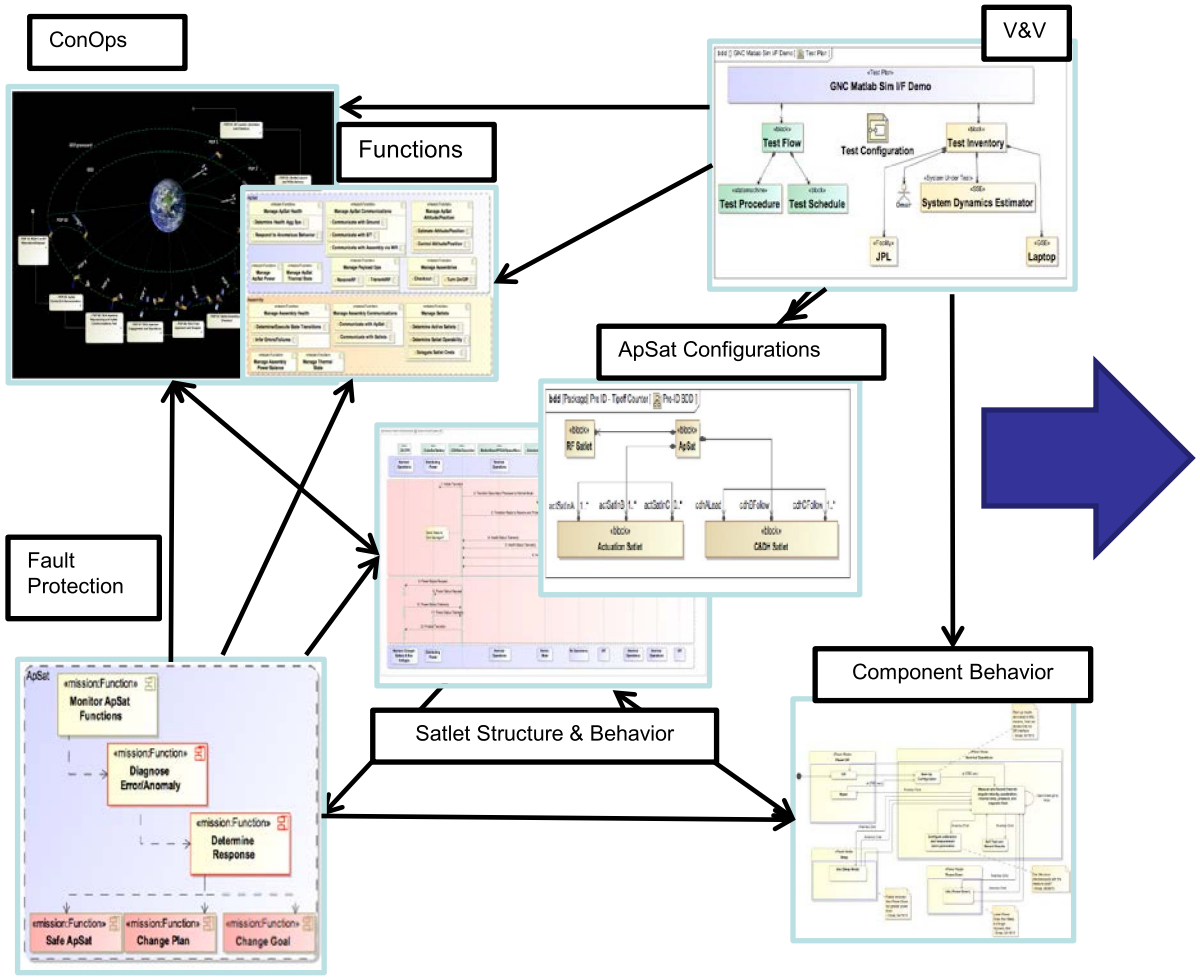


Satlet Structure & Behavior

Component Behavior



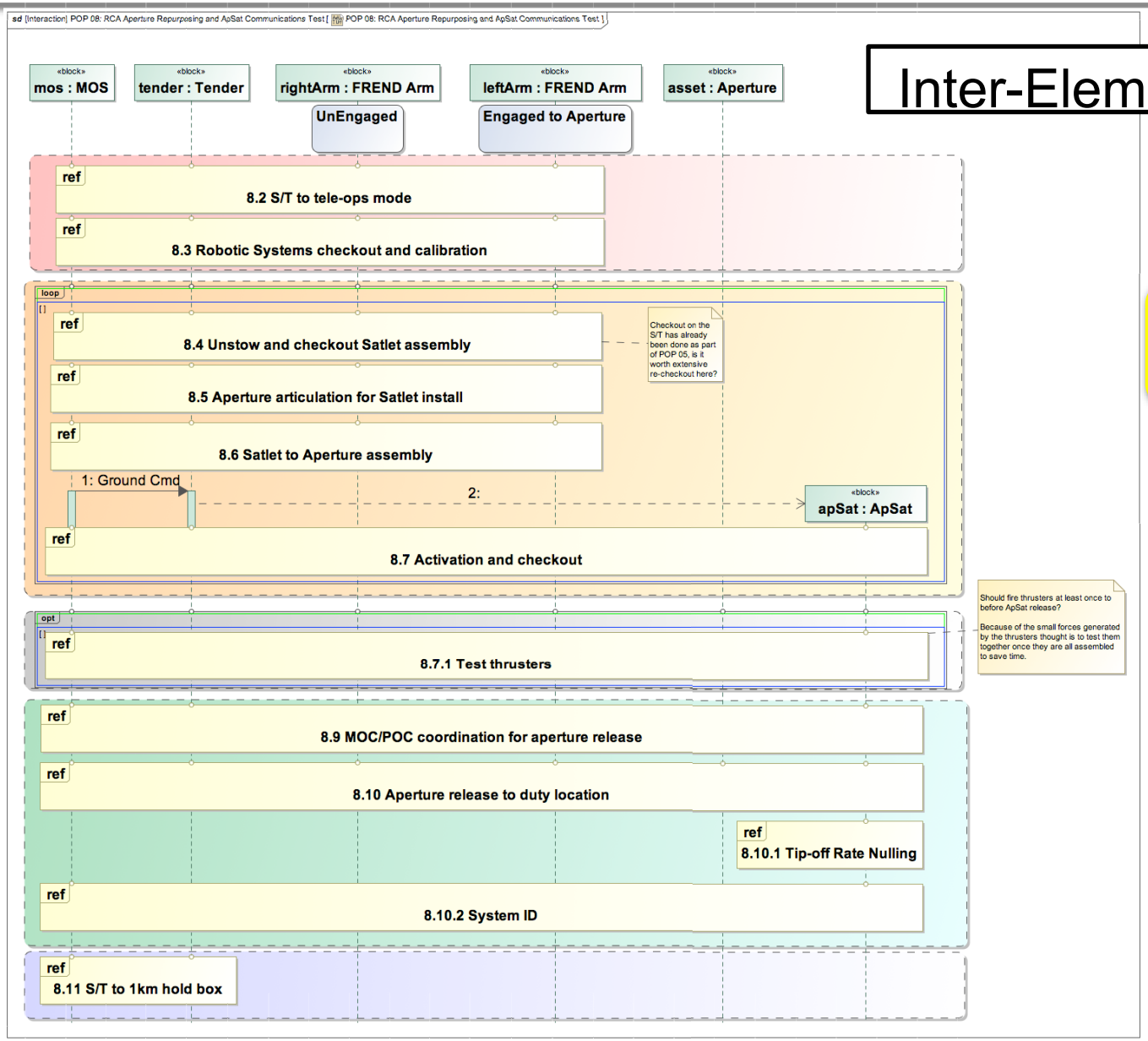
Model-Based Design Approach: Overview



Resulting Products

- Require-ments
- Detailed Design Specification
- Layered Software Architecture
- Fault Management Architecture
- Model-Based Design Verification: Check ConOps consistent with Component Behavior
- Test Plans
- Online System ID and Control Tuning

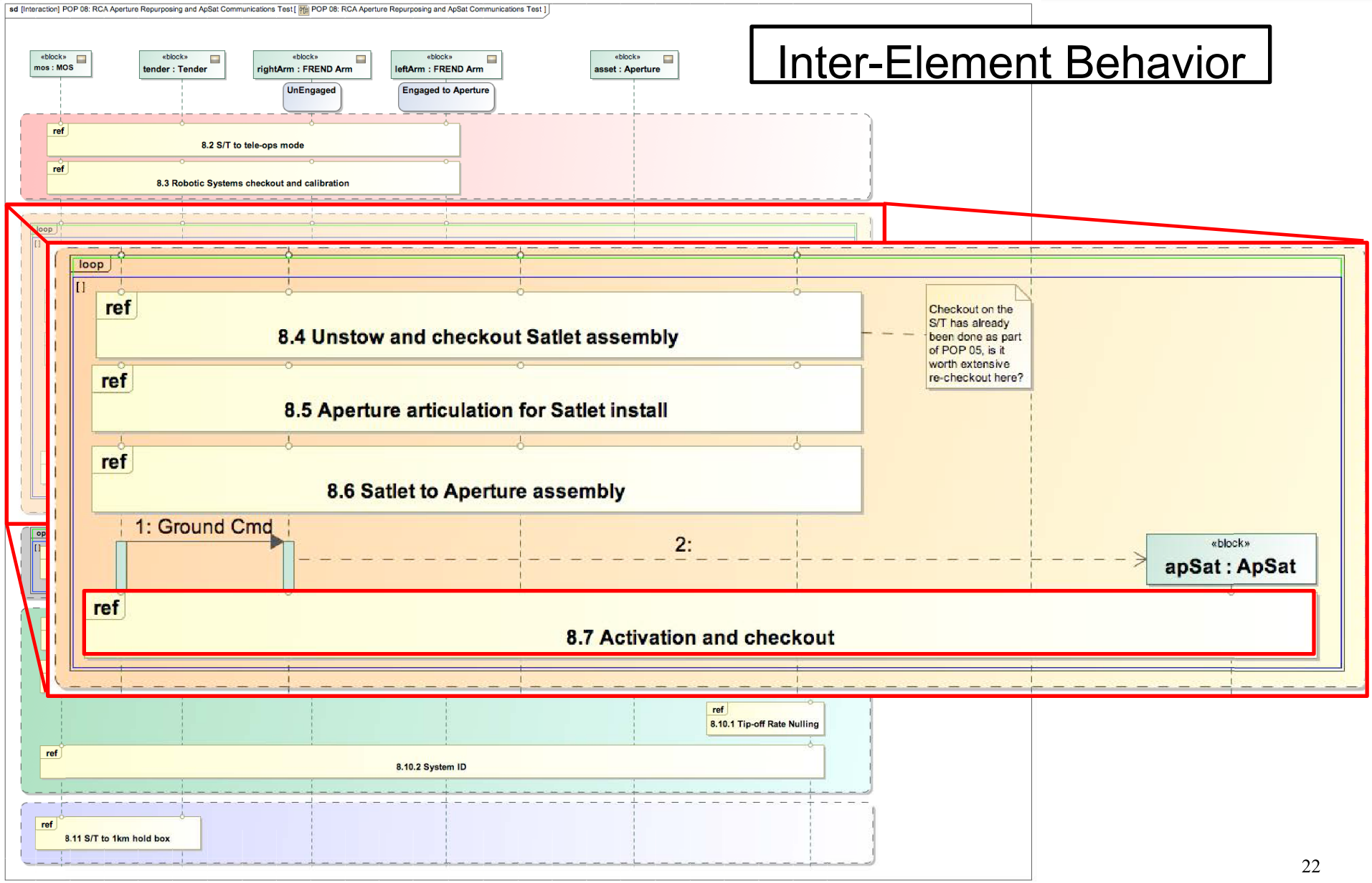
ConOps Modeling (Top-Down)



Inter-Element Behavior

Specify & Elaborate System-level Behaviors

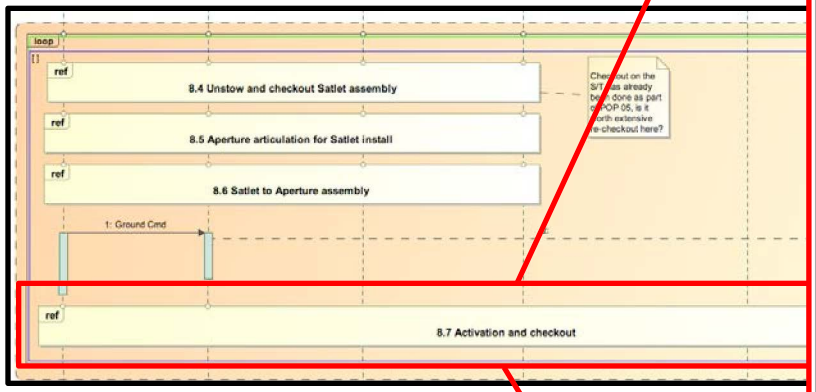
ConOps Modeling (Top-Down)

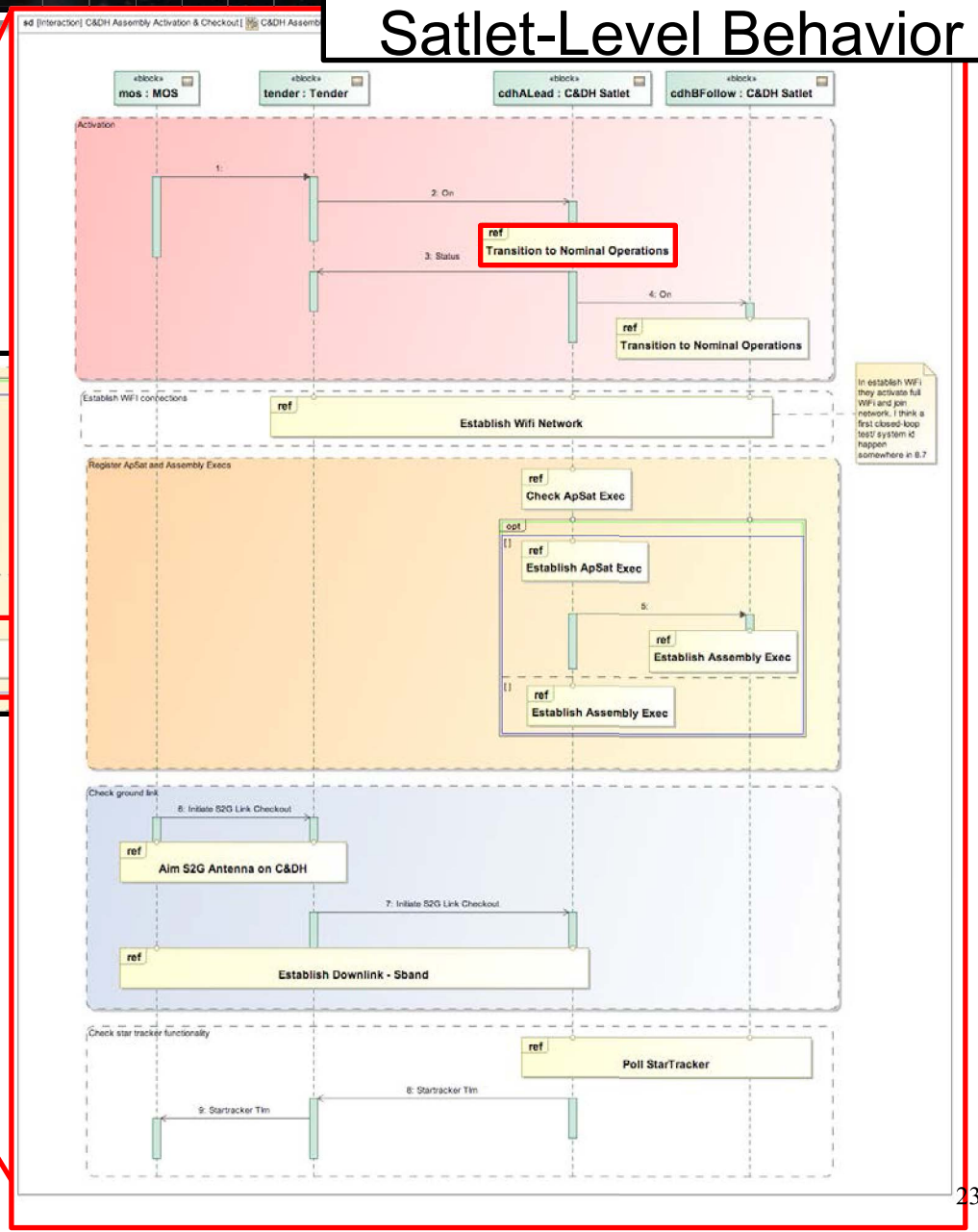


Further ConOps Elaboration (Top-Down)

Satlet-Level Behavior

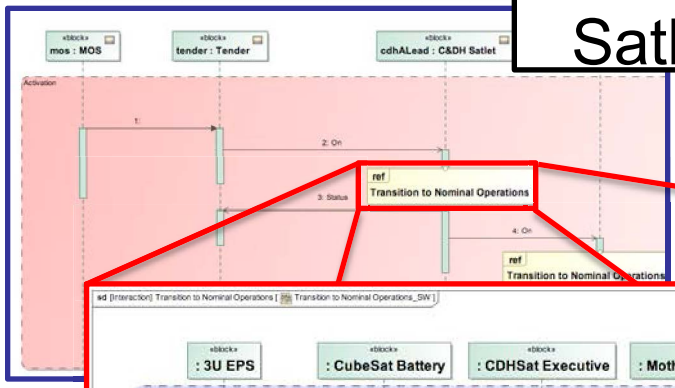
Inter-Element Behavior



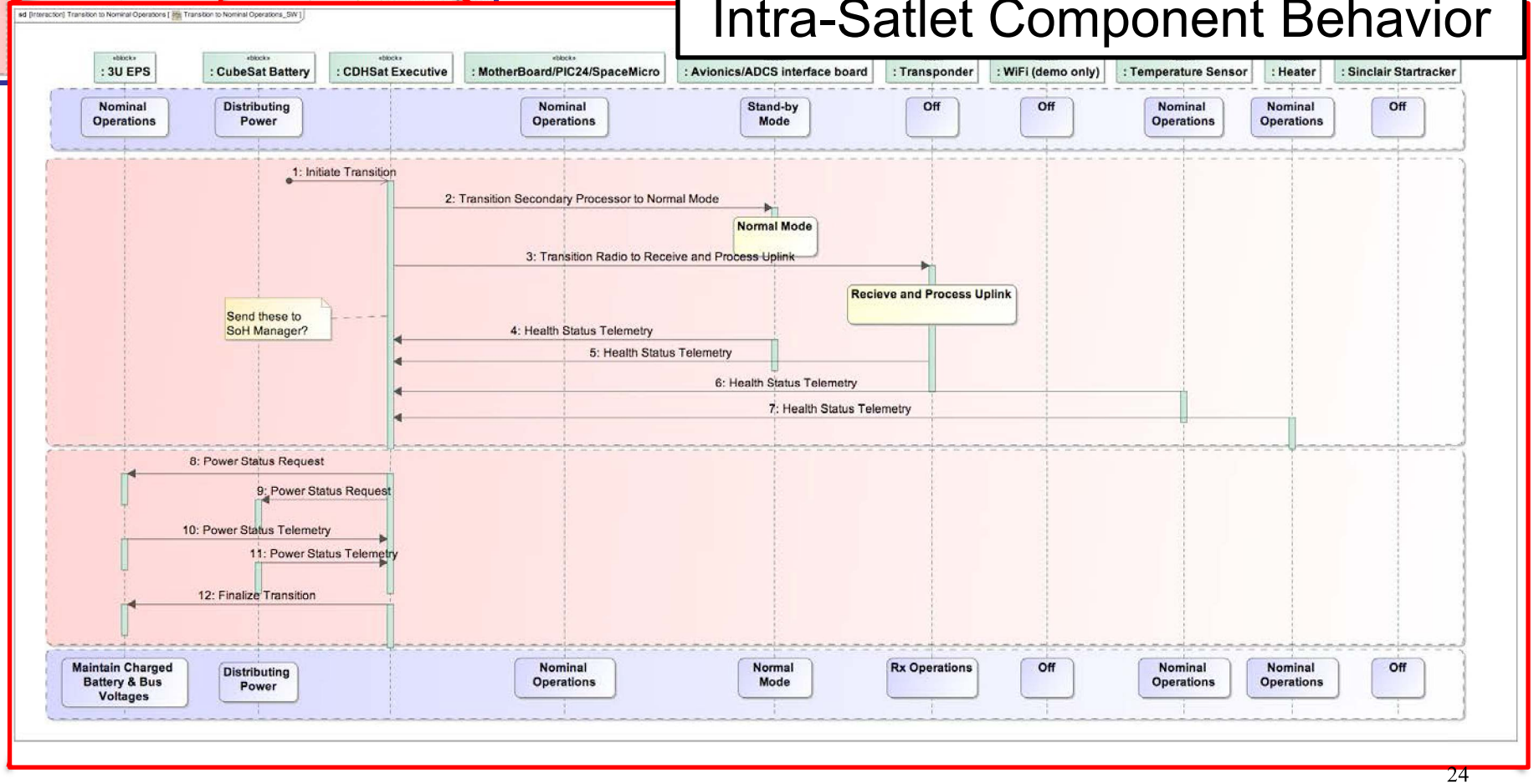


Further ConOps Elaboration (Top-Down)

Satlet-Level Behavior



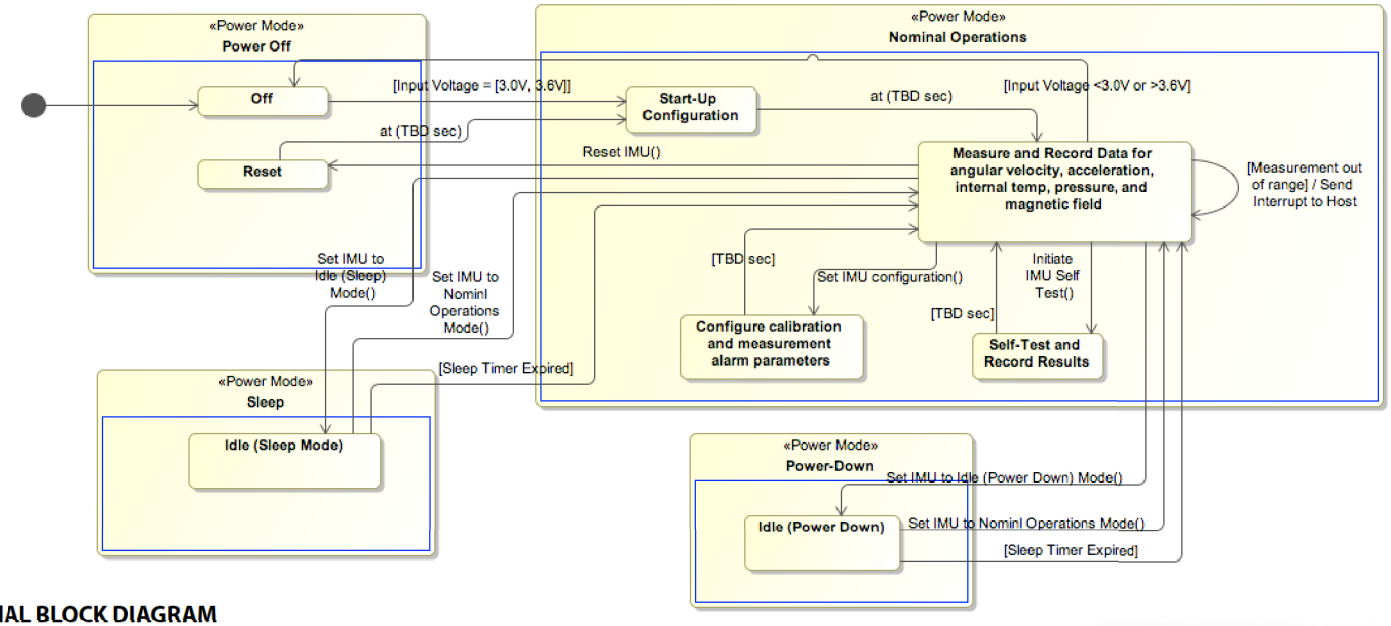
Intra-Satlet Component Behavior

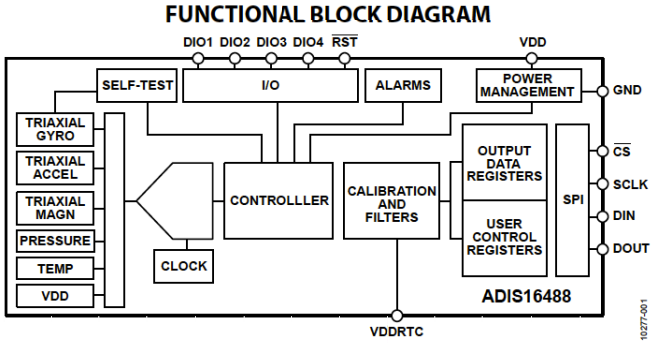


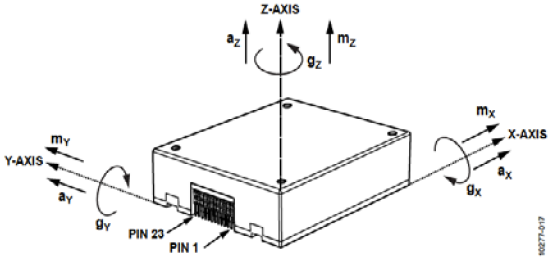
Component-Level Behavior Modeling (Bottom-Up)

- Component-level behavior captured in state charts based on component spec sheets.
- Can be used to develop FSW, FP, test plans, etc.

IMU



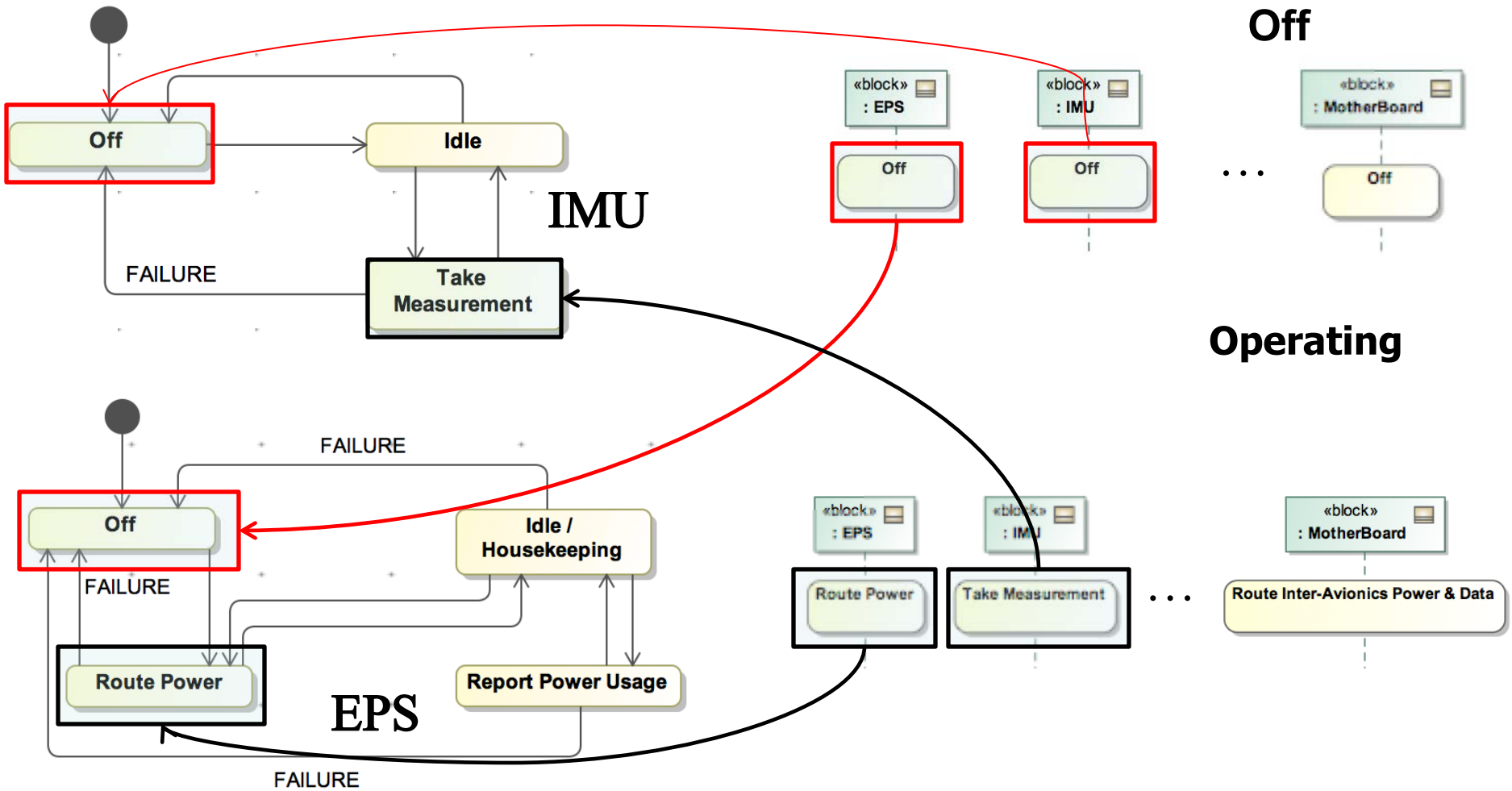




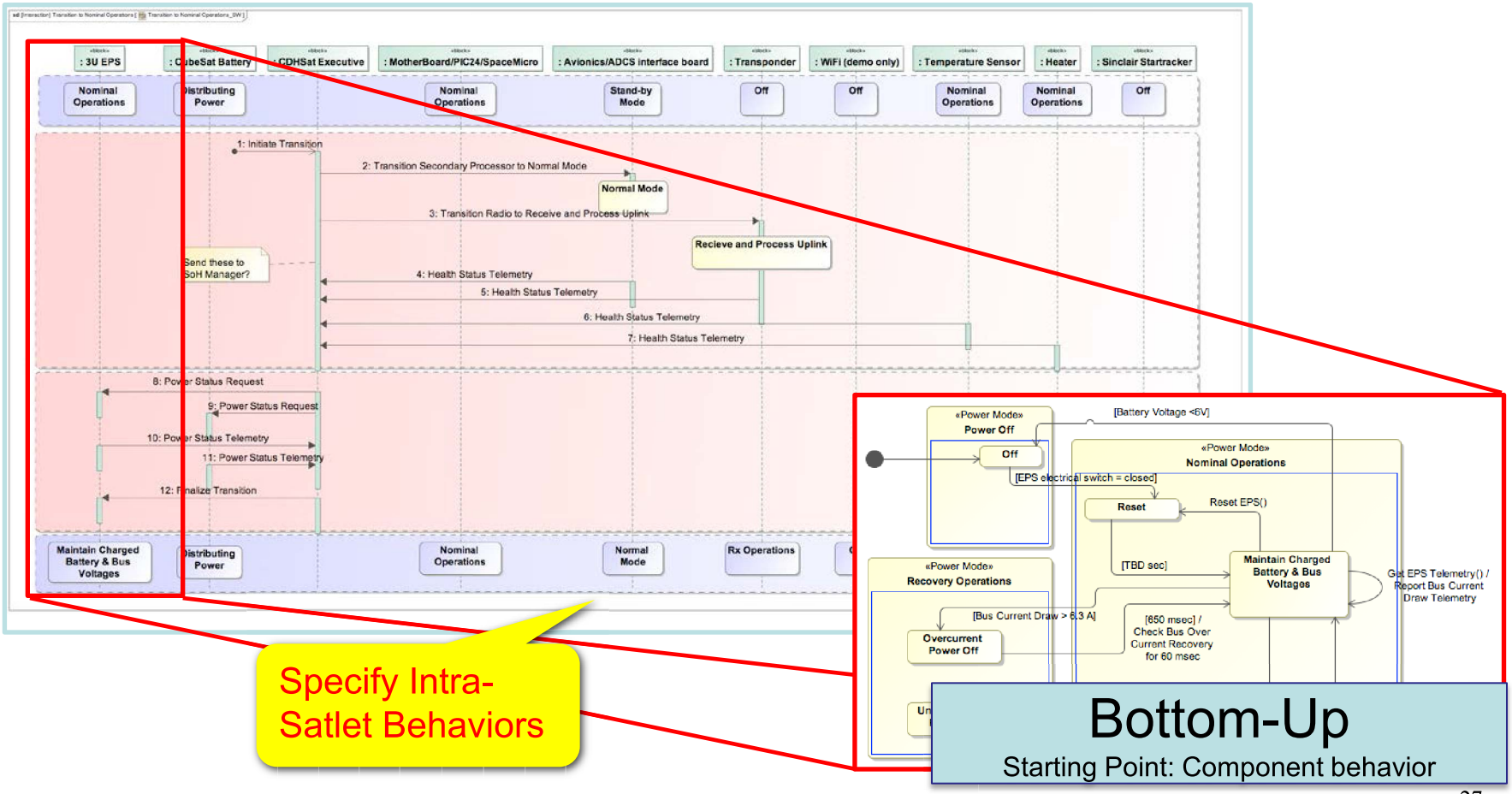
State Behavior Aggregation to Satlet Level (Bottom-Up)

Component-Level States

System "Modes"



State Behavior Aggregation to Satlet Level (Bottom-Up)

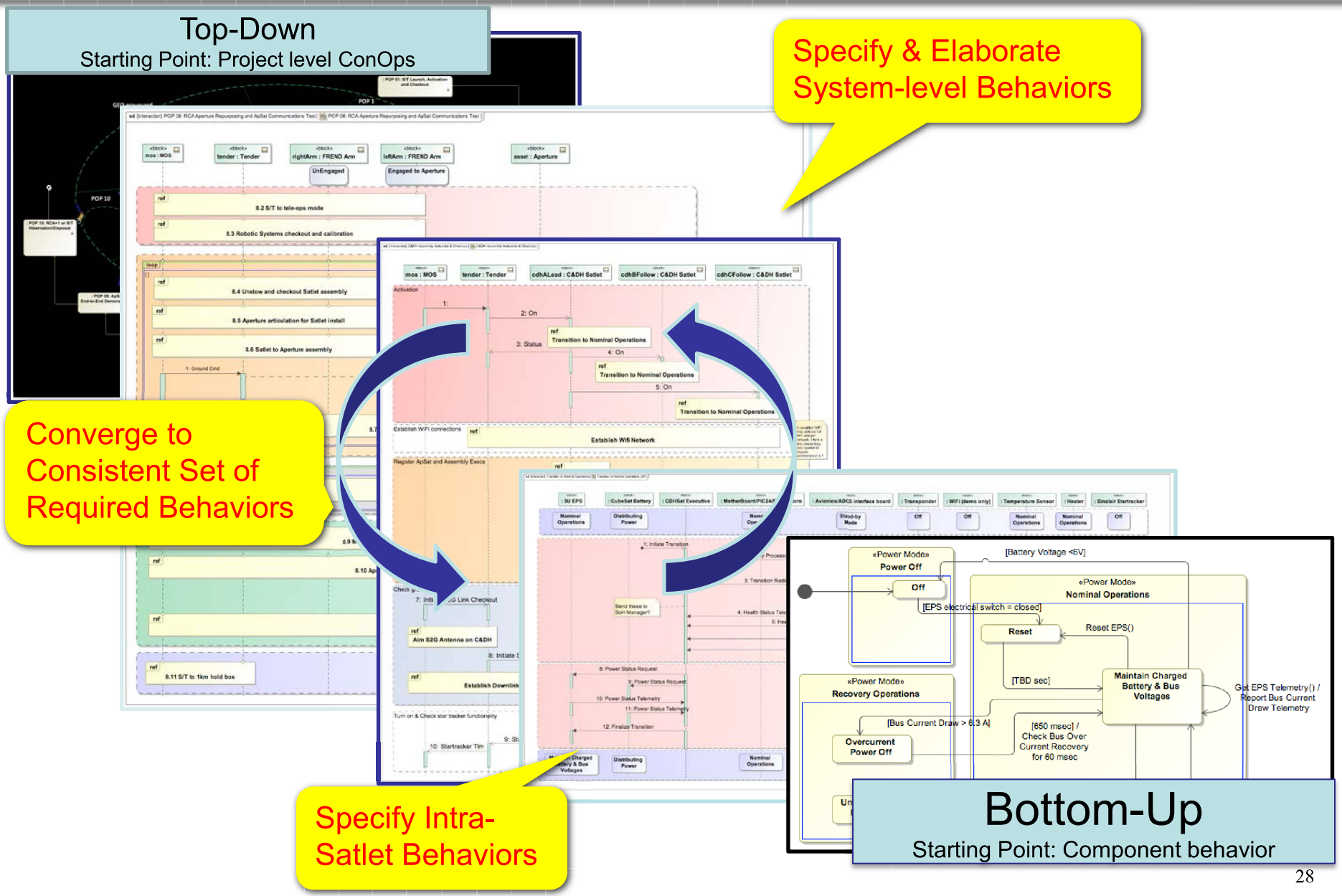


Specify Intra-Satlet Behaviors

Bottom-Up

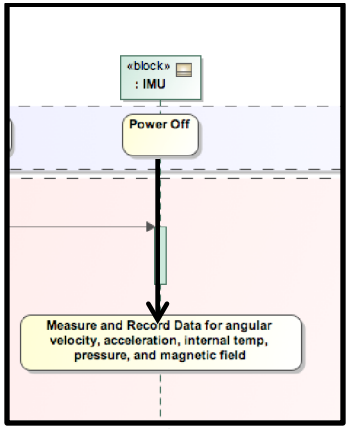
Starting Point: Component behavior

Model-Based Design ("Meet in the Middle")

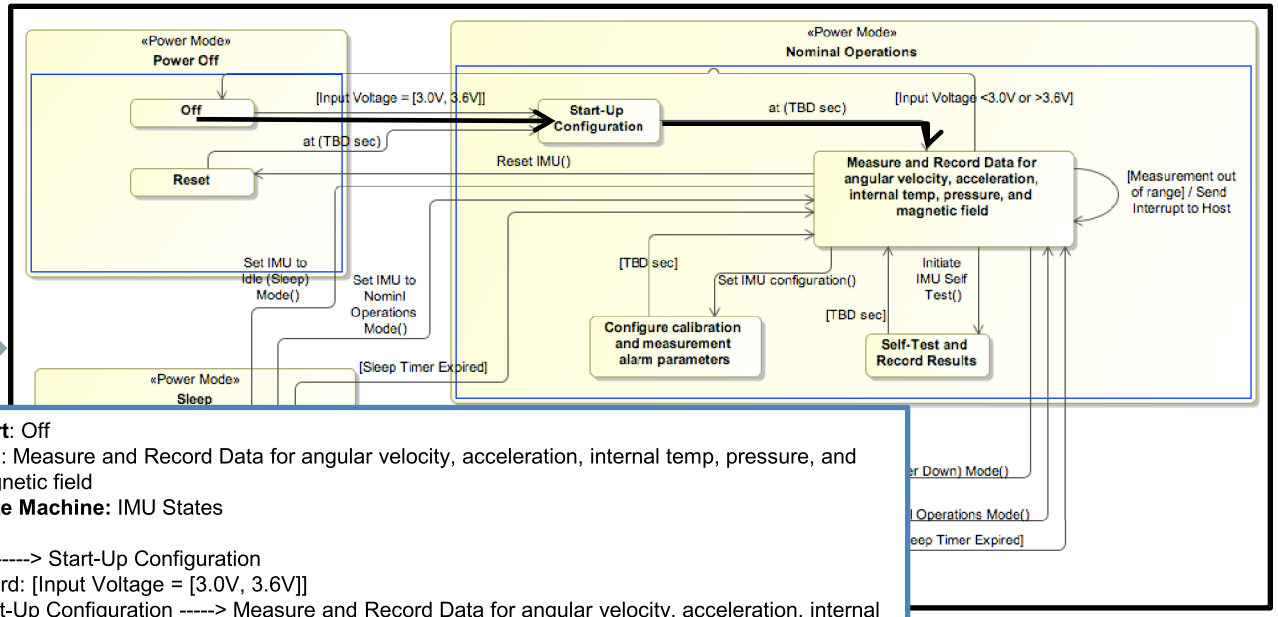


Behavior Verification ("Meet in the Middle")

Desired Sequences



- Need to verify that a specified sequence can be performed by the available components.
- Have both high-level sequences and component-level states



Start: Off
End: Measure and Record Data for angular velocity, acceleration, internal temp, pressure, and magnetic field
State Machine: IMU States

Off -----> Start-Up Configuration
Guard: [Input Voltage = [3.0V, 3.6V]]
Start-Up Configuration -----> Measure and Record Data for angular velocity, acceleration, internal temp, pressure, and magnetic field
Trigger (Time): at TBD sec

*****END STATE REACHED*****

Path: [Off,
Start-Up Configuration,
Measure and Record Data for angular velocity, acceleration, internal temp, pressure, and magnetic field]

Trace

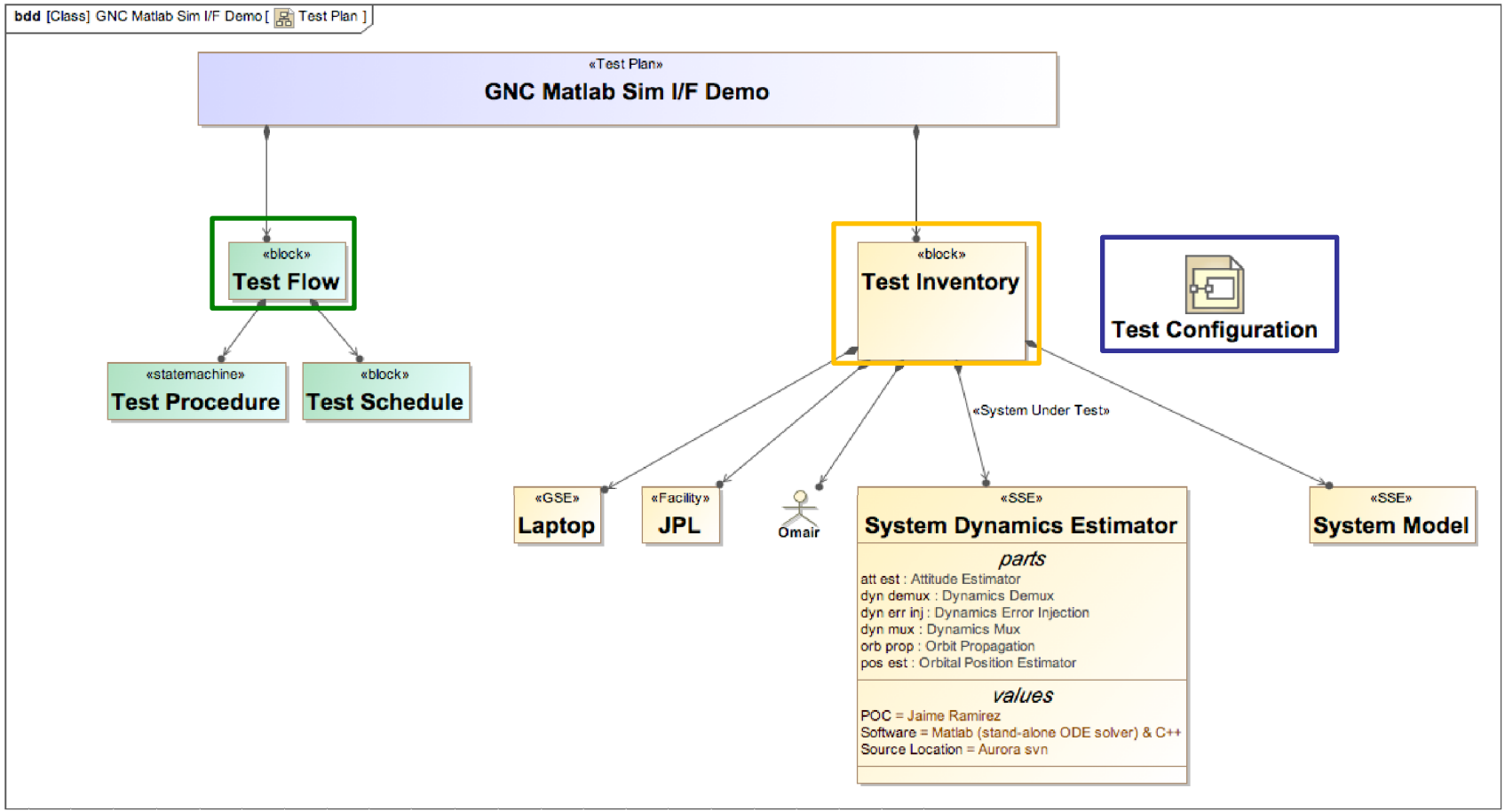
Component Behavior

Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

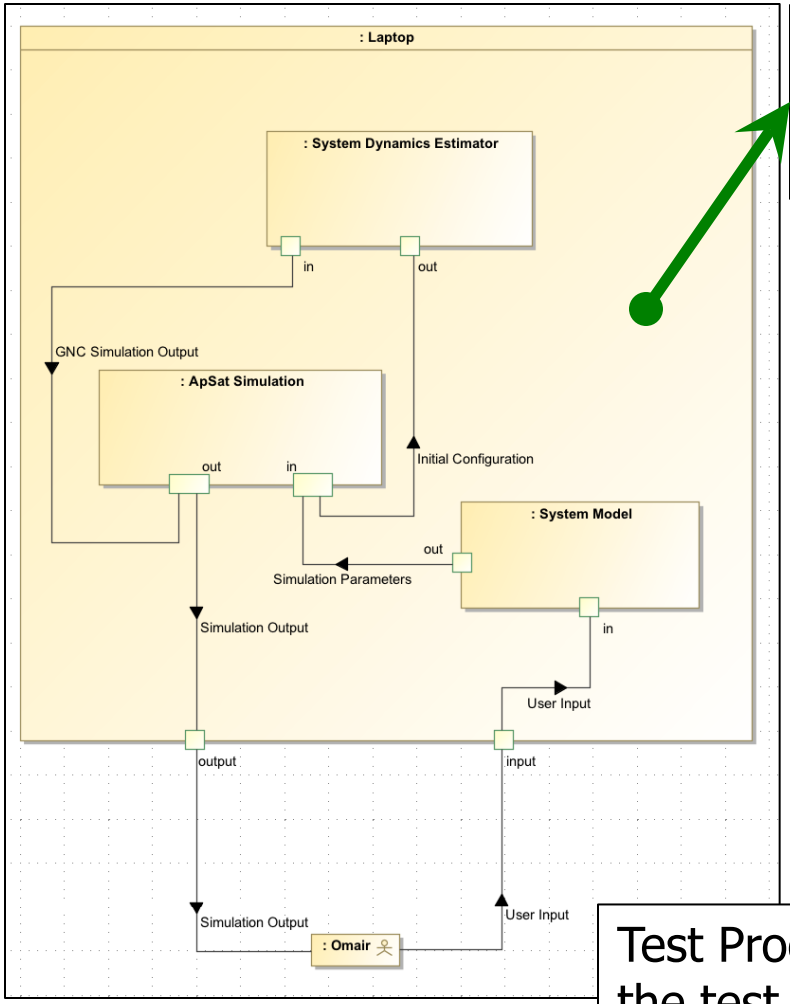
V&V Planning Leverages the System Model

- V&V products and planning greatly leverage the consolidated capture of design elements in the system model
- Test plans are being captured directly in the system model

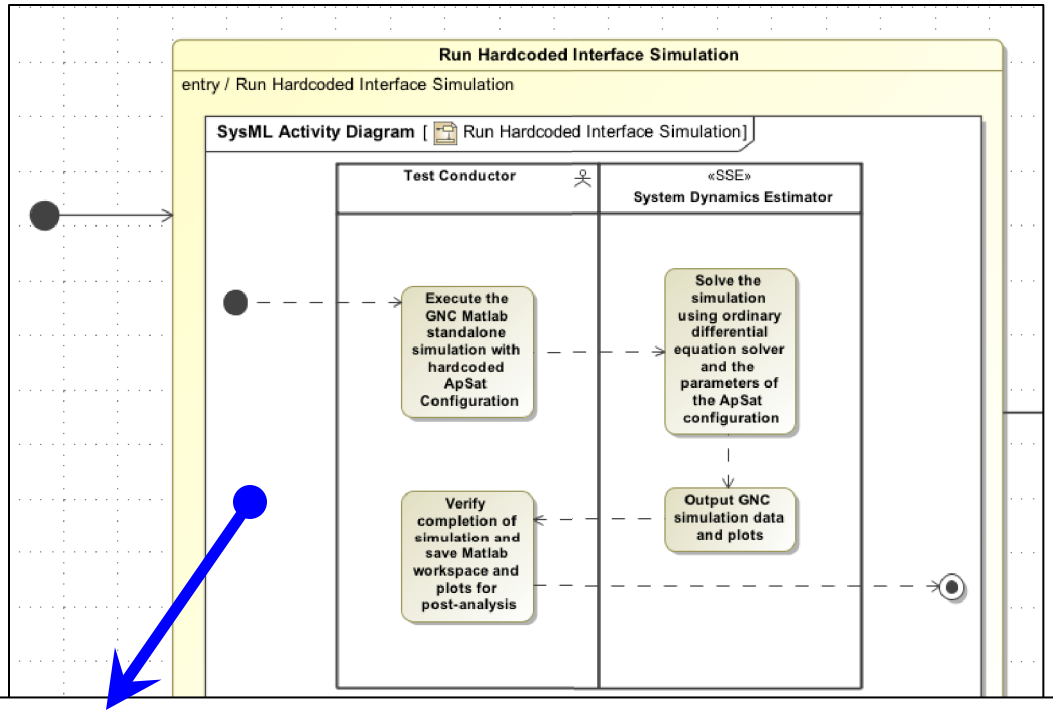


V&V Planning – Test Plans in the System Model

Test plans in the system model allow for efficient and consistent representation of test procedures and configurations.



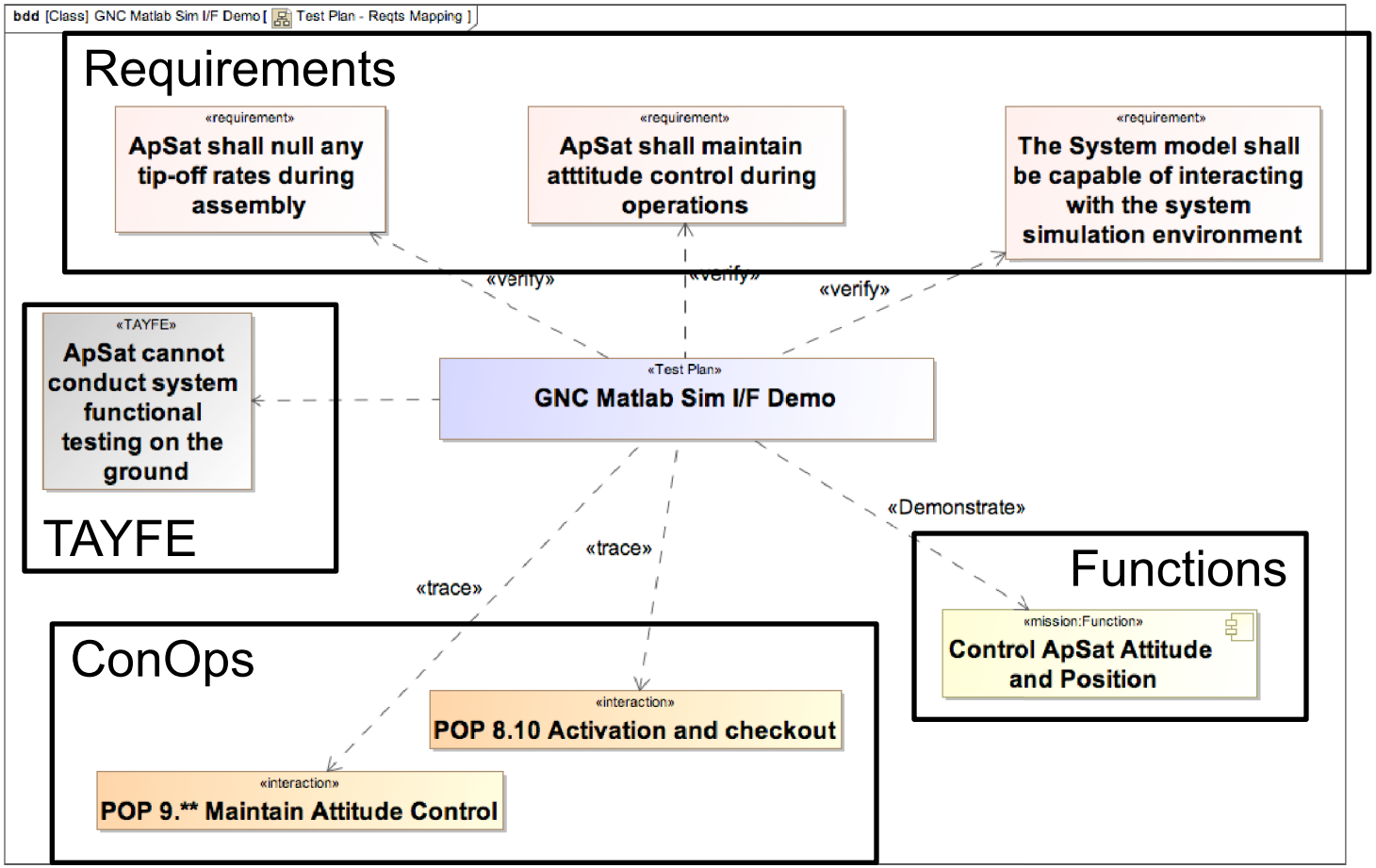
Test Configurations show the detailed mechanical, power, data, and optical connections and interfaces among elements in the test



Test Procedures capture the detailed steps for executing the test. Procedures are divided into swimlanes, which keep track of who/what is responsible for each step.

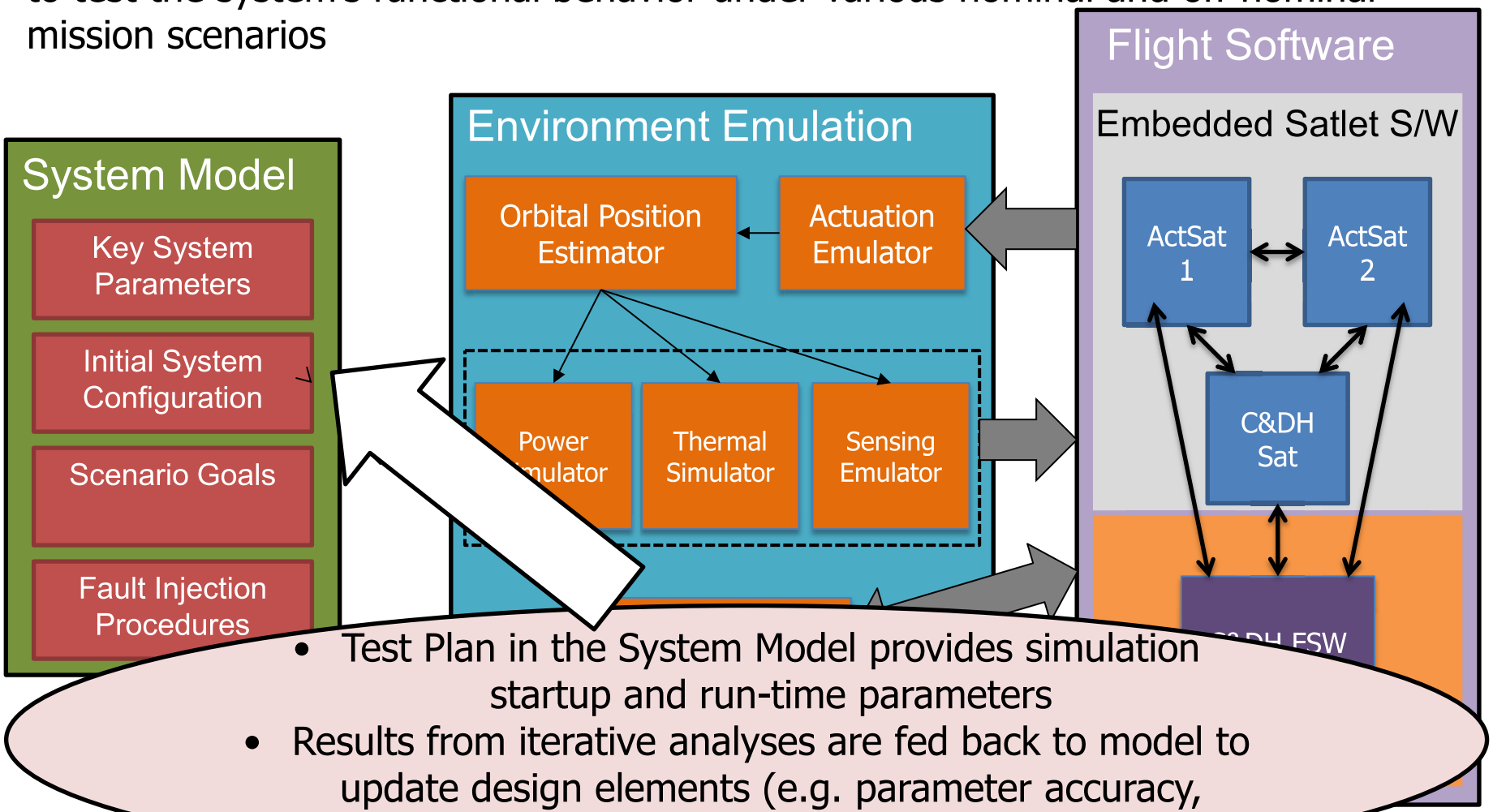
V&V Planning Leverages the System Model

- V&V products and planning can be traced to other elements in the system model



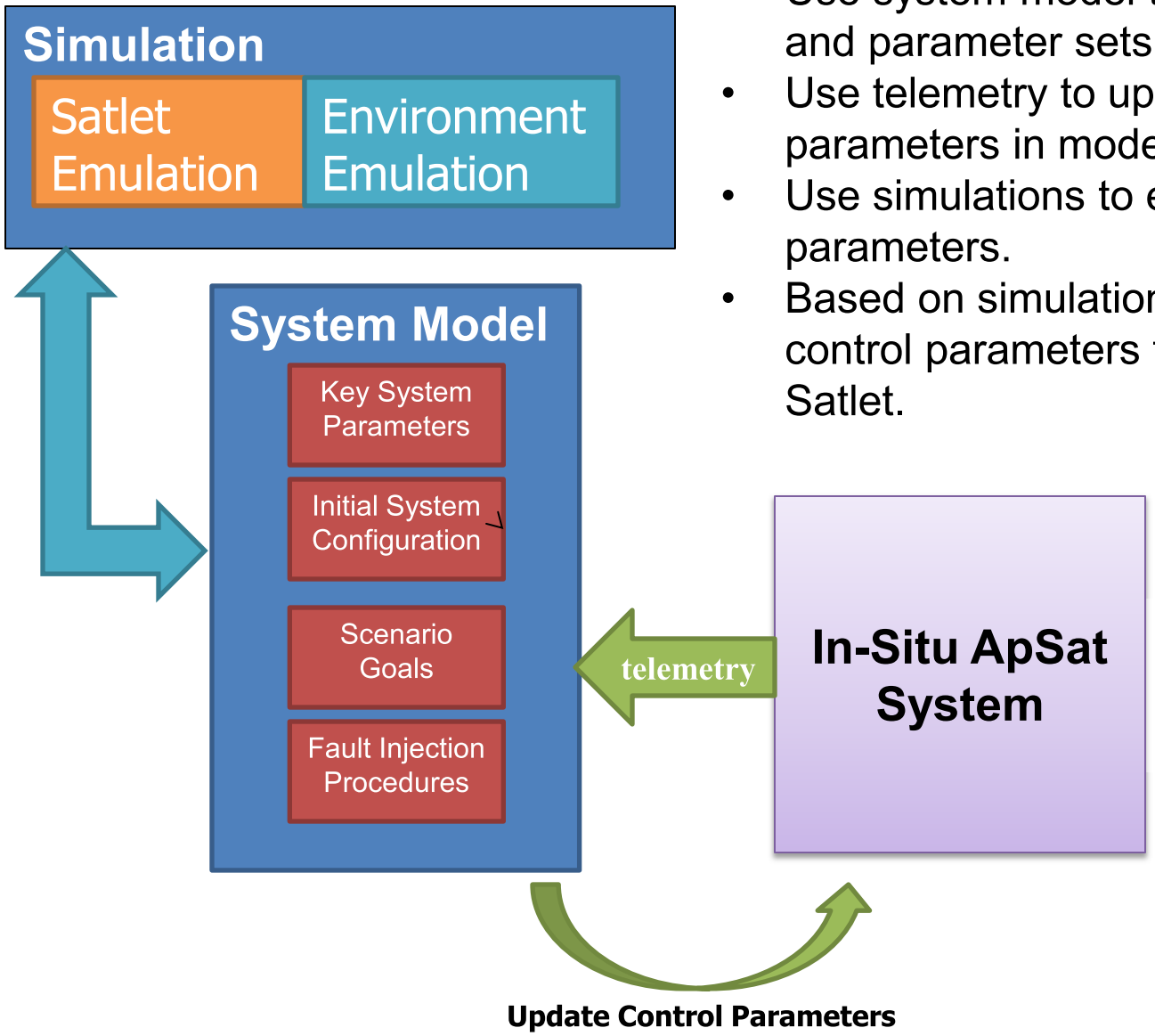
Model-Based V&V Architecture

The ultimate goal of Model-Based V&V is to have the three analysis elements (System Model, Environment Emulation, and Flight Software) working together to test the system's functional behavior under various nominal and off-nominal mission scenarios



- Test Plan in the System Model provides simulation startup and run-time parameters
- Results from iterative analyses are fed back to model to update design elements (e.g. parameter accuracy, ConOps efficiency and safety)

MBV&V Adaptation for System ID (during Ops)

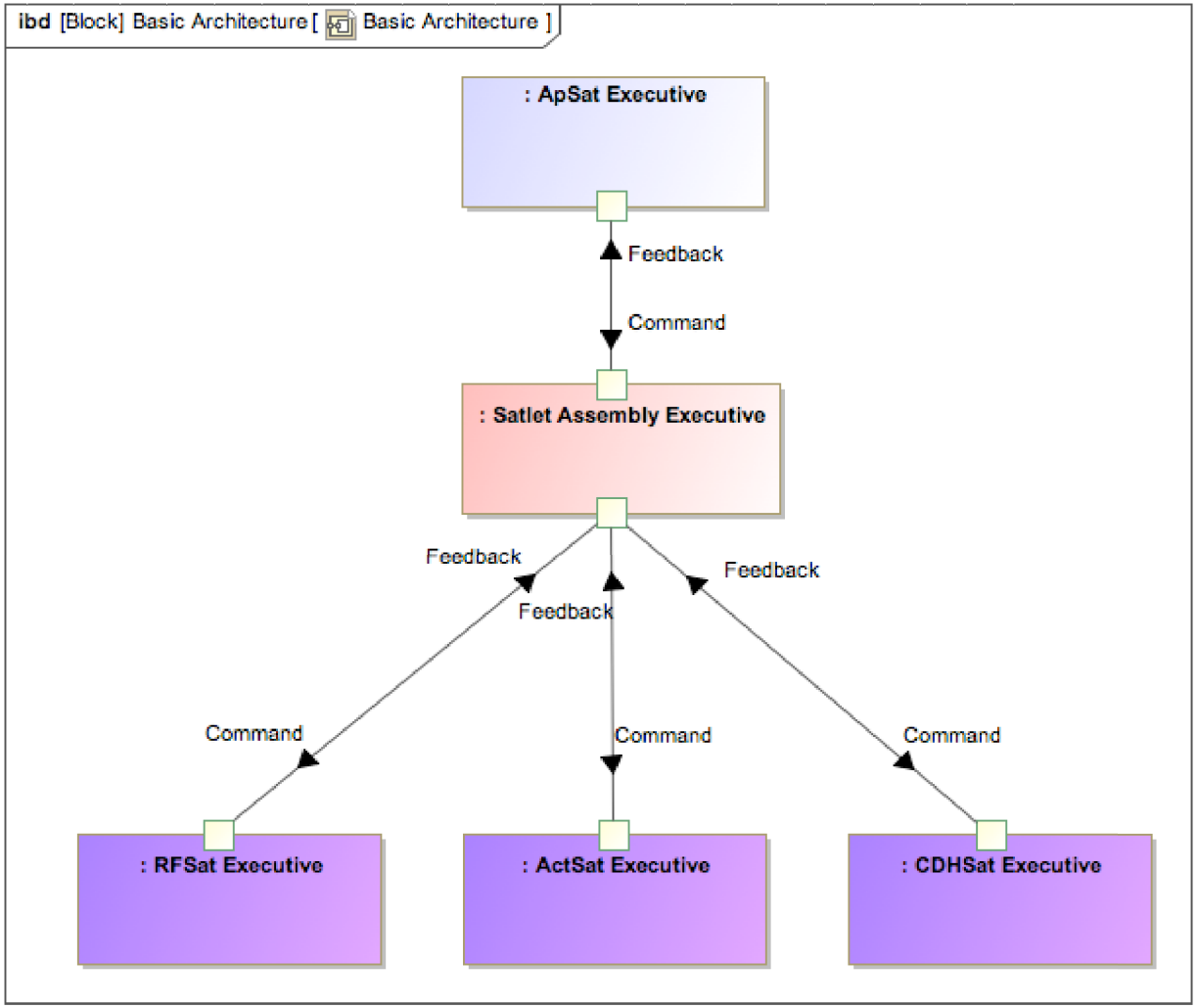


- Use system model to track configurations and parameter sets.
- Use telemetry to update key system parameters in model.
- Use simulations to estimate “un-measurable” parameters.
- Based on simulation results, update onboard control parameters for ApSat, Assembly or Satlet.

Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

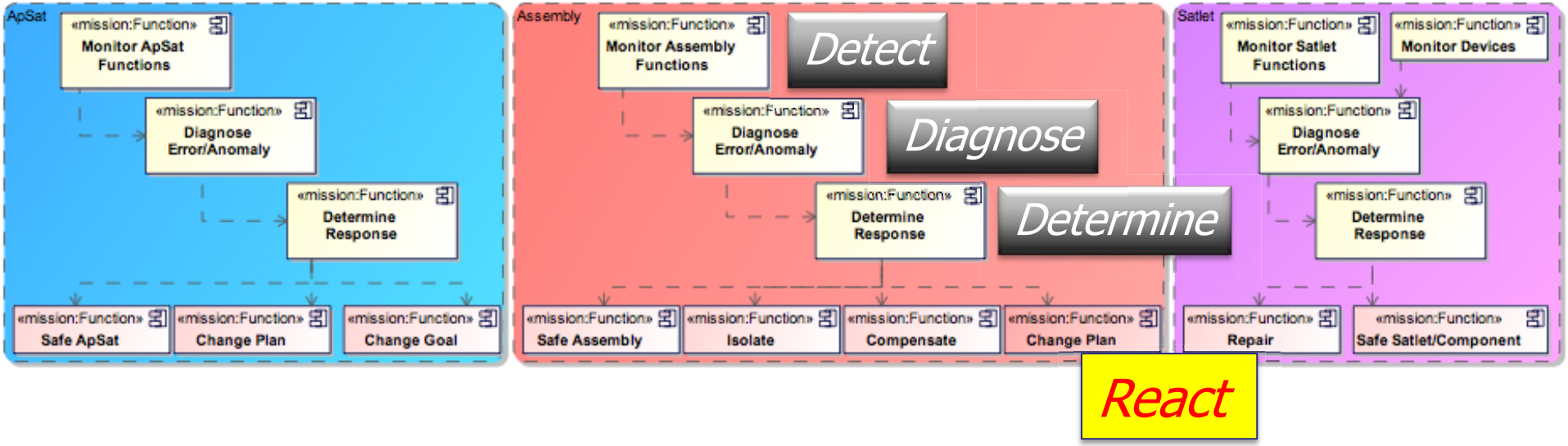
Flight Software Architecture



- Layered architecture, organized by aggregation level and functional domain
- Decision-making is distributed and computation is aggregated.
- Software design being captured in Satlet system model

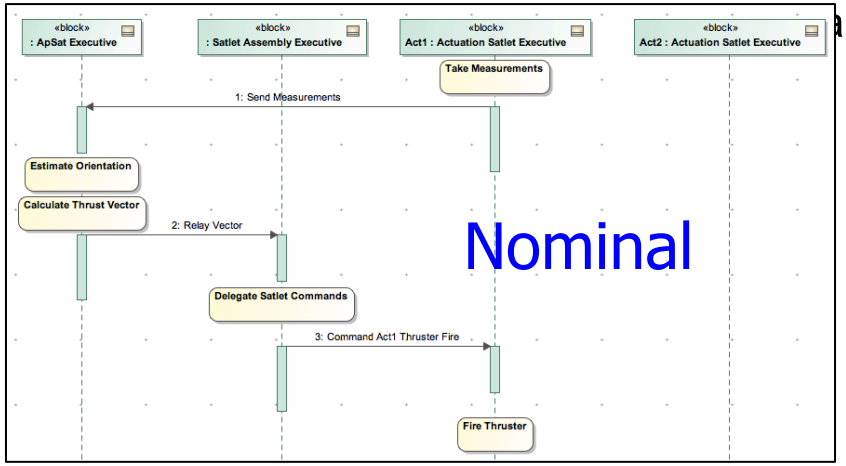
Fault Protection Design

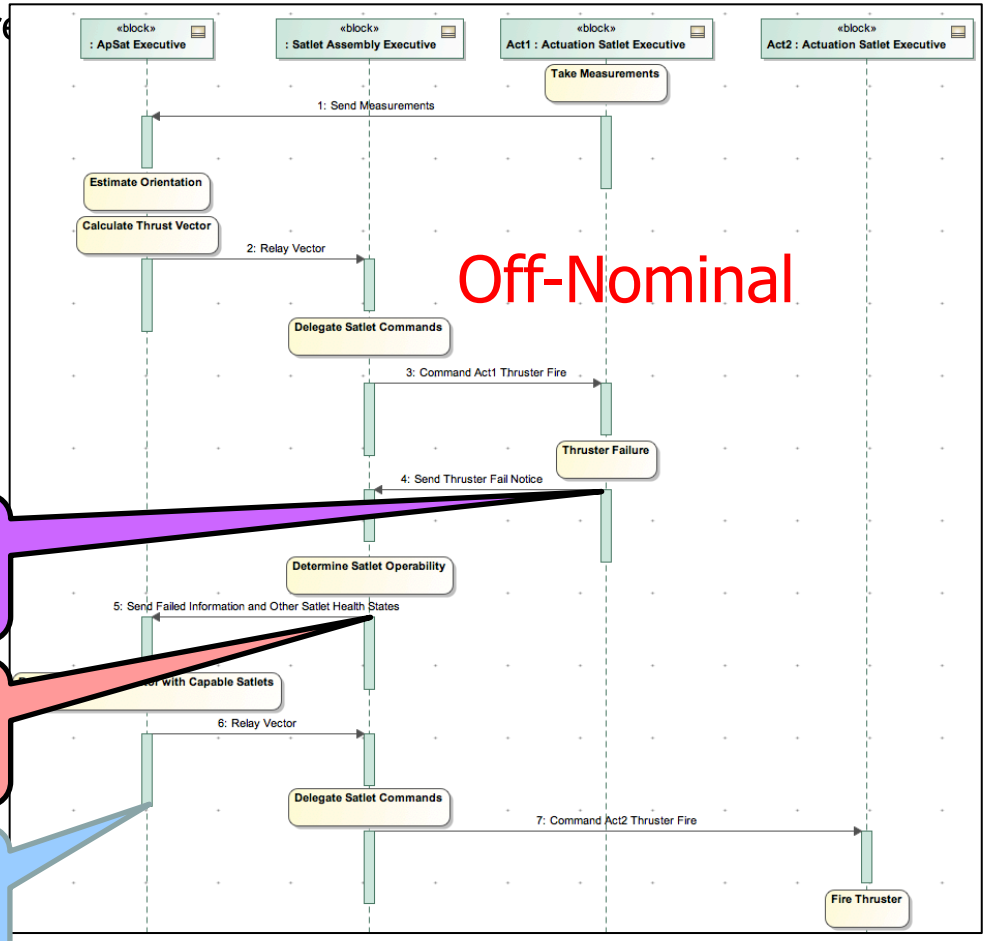
- **The redundancy and hierarchy in our cellularized system architecture can be leveraged for system robustness**
  - Each layer monitors its functions, only the Satlet layer monitors devices
  - Different types of responses are possible at different layers
  - Enable local reactions to problems and fault masking
- **Model-based systems engineering approach provides foundation for FP architecture design**
  - Mission success tree is directly related to ConOps, outlines most critical functions
  - In development: auto-generation of FMECA, to explain failure modes, causes, effects
  - Behavior is specified and validated using system model, then embodied in FSW



Off-Nominal Behavior – Thruster Failure Example

- **Sequence diagrams can be used to show both nominal and off-nominal behaviors, referring to states from relative state machine diagrams**
  - Note: Off-nominal sequences may end in the nominal state or may end in some failure





Actuation Satlet Response:
Safe Satlet, Report failure to Assembly, Attempt repair (reset thruster electronics)

Assembly-level Response:
Isolate failed Satlet if it is drawing Assembly power, Report loss of capability to ApSat

ApSat-level Response:
Update state with reported loss of capability, issue new cmds to Assemblies to compensate

Outline

- Background on the DARPA Phoenix Program
- Development Process Overview
- Overview of Aurora/JPL/MIT Satlets Architecture
- Model-based Design Approach
- Model-based V&V Approach
- Other Products
- Conclusions

Conclusions

- A Model-Based Design Approach allows management of the flexibility and complexity associated with a cellularized architecture.
- Different configurations can be represented and tracked; this information can be used to synthesize trade space analyses and seed what-if scenarios.
- Provides an integrating mechanism for top-down and bottom-up design to "meet in the middle", enabling an iterative design process that converges on the desired behavior and enabling consistency to be checked.
- System model scope spans the system design from hardware specifications to software specifications to ConOps, and provides a robust framework for planning and executing V&V in a formal and consistent manner.
- V&V is further enhanced by the ability to make completeness checks within the model and to output parameters that feed the testing (model-in-the-loop testing).