# Executable Statemachines in Cameo Simulation Toolkit

Nerijus Jankevicius

June 16, 2015, PSSM meeting, Berlin


magicdraw
Architecture Made Simple
magicdraw
+
CAMEO
SIMULATION TOOLKIT™
CAMEO
SIMULATION TOOLKIT™
CAMEO™
SUITE

# Customers

- More than 150 companies in 5 years
  - NASA/JPL, Lockheed Martin, Bombardier Transportation, Aerospace Corporation, MITRE, General Dynamics, US Army, Siemens, BEA Systems, Microsoft/Nokia, ESO and others…

# Cameo Simulation Toolkit

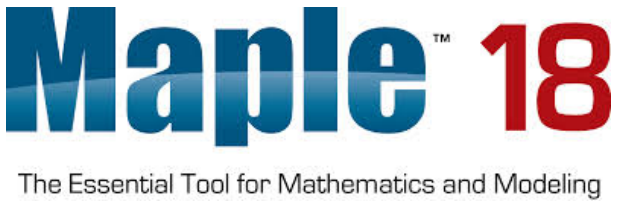
- Model execution framework and infrastructure:
  - Model debugging and animation environment
  - Pluggable engines, languages and evaluators
  - User Interface prototyping support
  - Model driven configs and test cases

- 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)

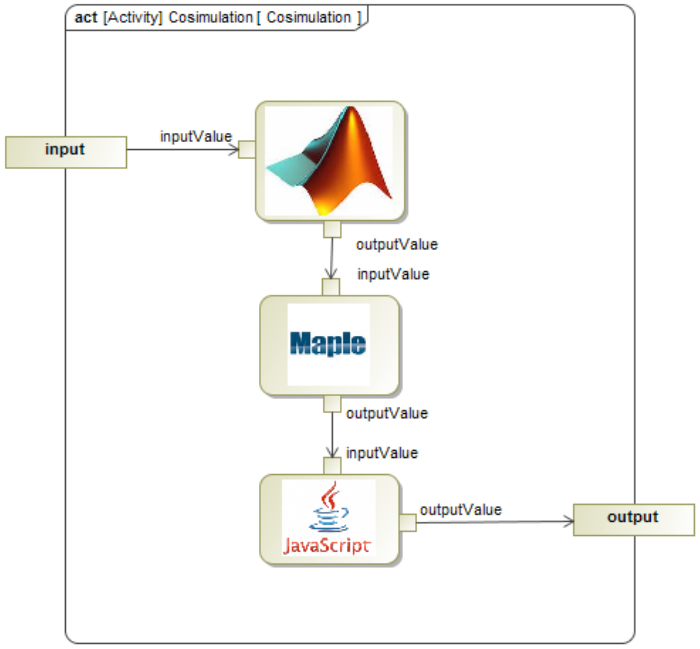


# Integration of analytical models



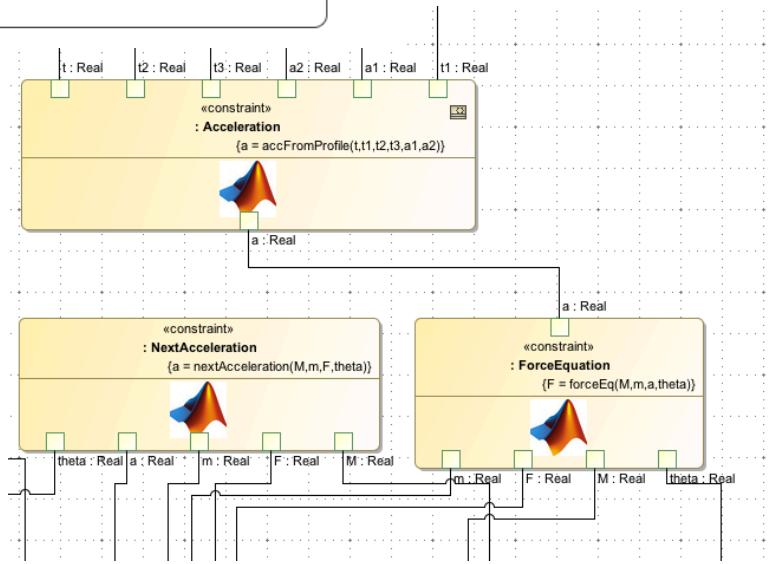
# Action languages

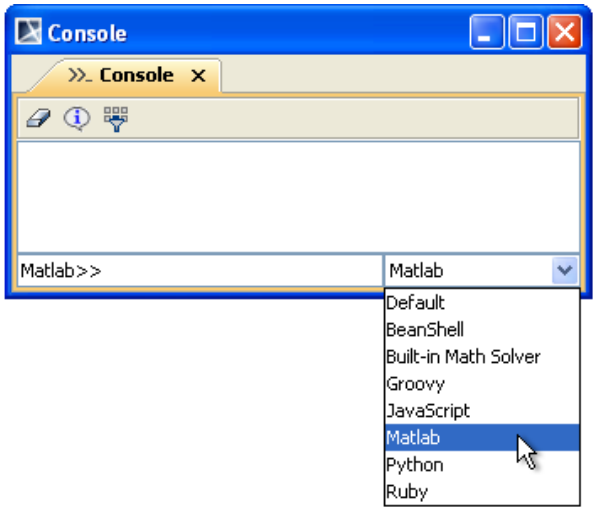



Math engines

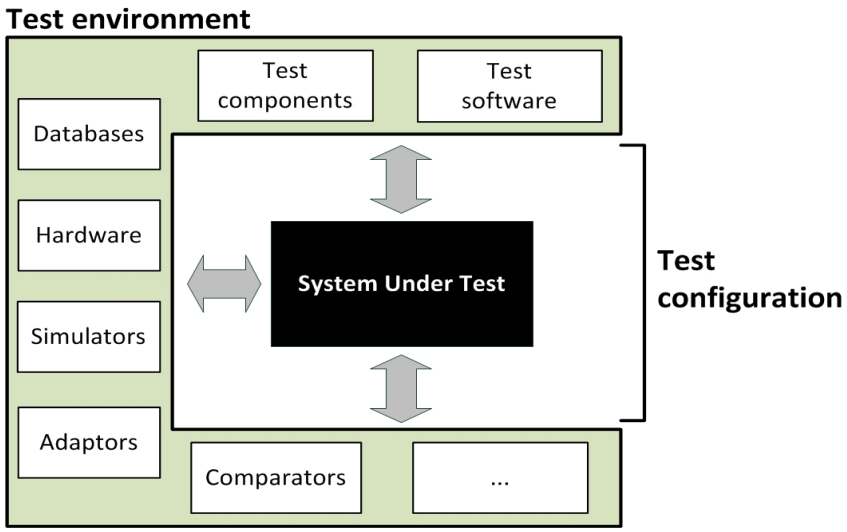
- Matlab/Simulink
- Mathematica
- Maple
- Open Modelica

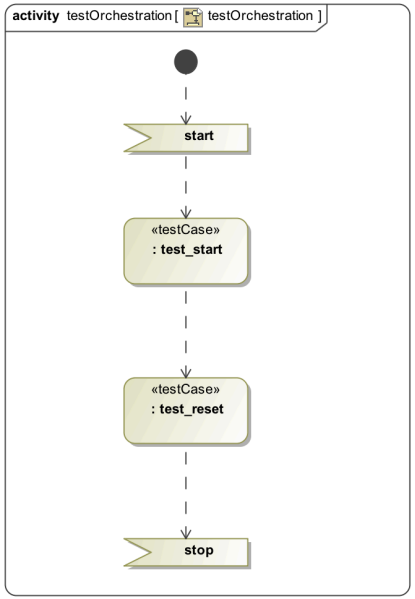
- Scipting
  - Javascript
  - Python
  - Groovy
  - Ruby

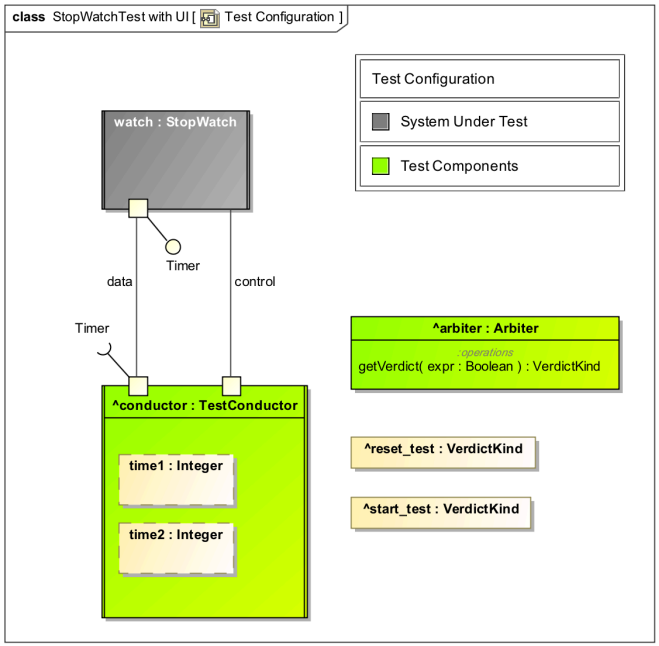


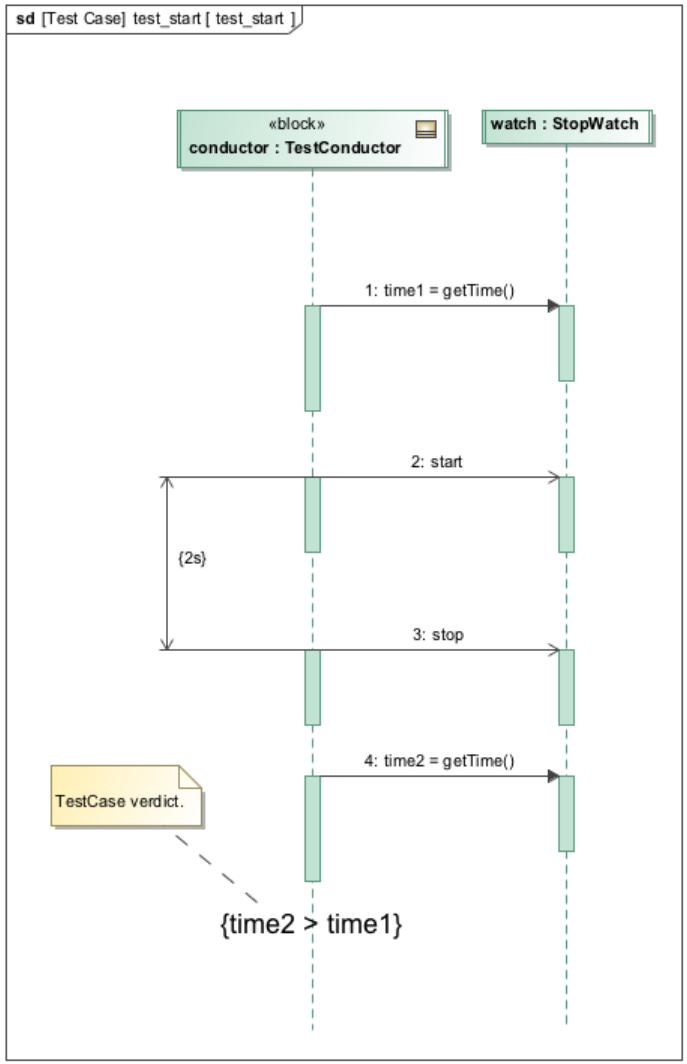



# Model-based testing, UTP

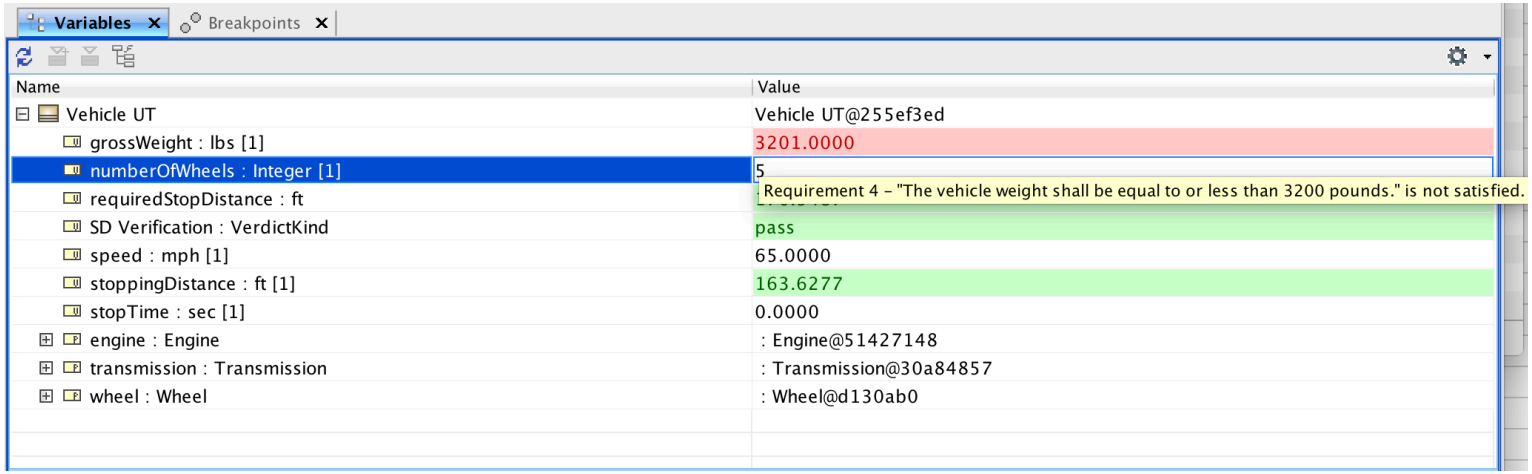


# Requirements verification




| # | Name | Gross Weight : Lbs | Speed : Mph | Number Of Wheels : Integer | Stopping Distance : Ft | Required Stop Distance : Ft | SD Verification : Verdict Kind | : Weight Constraint | : SD Constraint |
|---|---|---|---|---|---|---|---|---|---|
| 1 | vh1 | 2800.0 | 65.0 | 4 | 178.9119253179637 | 176.54869701553358 | fail | pass | fail |
| 2 | vh2 | 3300.0 | 65.0 | 5 | 168.68838672836577 | 176.54869701553358 | pass | fail | pass |
| 3 | | 3200.0 | 65.0 | 4 | 204.4707717919585 | 176.54869701553358 | fail | pass | fail |
| 4 | | 2700.0 | 65.0 | 4 | 172.522213699465 | 176.54869701553358 | pass | pass | pass |
| 5 | | 2700.0 | 65.0 | 4 | 172.522213699465 | 176.54869701553358 | pass | pass | pass |
| 6 | vehicle UT | 3500.0 | 65.0 | 4 | 223.6399066474546 | 176.54869701553358 | fail | fail | fail |
| 7 | vehicle UT1 | 2700.0 | 65.0 | 4 | 172.522213699465 | 176.54869701553358 | | pass | pass |

| 2 | Gross Weight : Lbs | Speed : Mph | Number Of Wheels : Integer | Stopping Distance : Ft | Required Stop Distance : Ft | SD Verification : Verdict Kind | : Weight Constraint | : SD Constraint |
|---|---|---|---|---|---|---|---|---|
| 3 | 2800 | 65 | 4 | 178.9119253 | 176.548697 | fail | pass | fail |
| 4 | 3300 | 65 | 5 | 168.6883867 | 176.548697 | pass | fail | pass |
| 5 | 3200 | 65 | 4 | 204.4707718 | 176.548697 | fail | pass | fail |
| 6 | 2700 | 65 | 4 | 172.5222137 | 176.548697 | pass | pass | pass |
| 7 | 2700 | 65 | 4 | 172.5222137 | 176.548697 | pass | pass | pass |
| 8 | 3500 | 65 | 4 | 223.6399066 | 176.548697 | fail | fail | fail |
| 9 | 2700 | 65 | 4 | 172.5222137 | 176.548697 | | pass | pass |

# State Chart XML - SCXML

## W3C

- http://www.w3.org/TR/scxml/
- W3C Proposed Recommendation 30 April 2015
- Provides a generic state machine–based execution environment based on Harel state charts.
- Is able to describe complex state-machines, including sub-states, concurrency, history, time events and more.
- Algorithm for SCXML Interpretation (Informal, Lisp-like language)
- Standard requires 2 reference implementations
- W3C provides test suite to test compliance with standard

# fUML and SCXML integration

**The problem**:

- fUML does not support Statemachines

**The solution**:

- Implement UML Statemachine to SCXML mapping
- wrap SCXML into fUML Execution
- fUML signalInstance to SCXML event
- fUML Activity Execution wrapped to SCXML <invoke>
- SCXML context -> fUML Object structural features

**The result**:

- Engines are independent, integration is part of the toolkit
- Statemachines as classifier behaviors
- Activities or OpaqueBehaviors as entry/do/exit behaviors of the State
- SendSignalAction can trigger Transitions in the Statemachine
- UML Statemachine can be exported to SCXML file

# fUML extensions

- EventOccurence as general class for all events.  Event pool modification.
  - Signal event
  - Time event
  - Change event
  - Call Event

- Add active state configuration to Object activation
- Multi-threading
- Event attribute values -> Effect, Entry/do/exit parameters
- OpaqueExpressions in guards

- ALF extensions?
  - Need for  inState(), getState(),  getCurrentEvent()
  - Qualified names of the states (State1::regionName::State2::regionName2::State3)

# Action Language Helper (ALH)

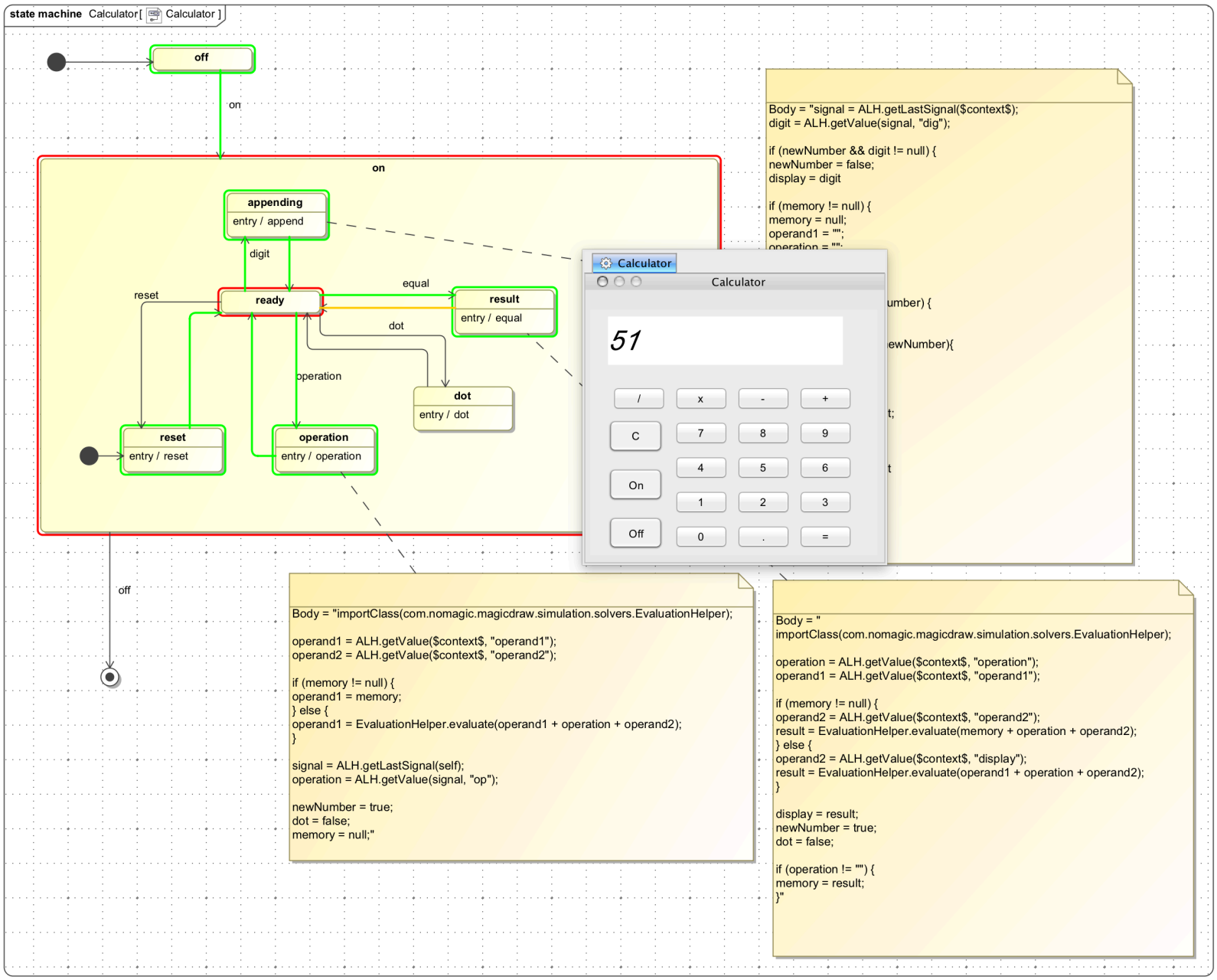
- Temporary alternative to real action language

- Java API to deal with fUML runtime Objects
  - Get context object (this, self)
  - Retrieve current event
  - Get token value
  - Set/get value of structural feature
  - Create object
  - Send signal
  - Call operation
  - Call/start behavior
  - Check if object is in particular state

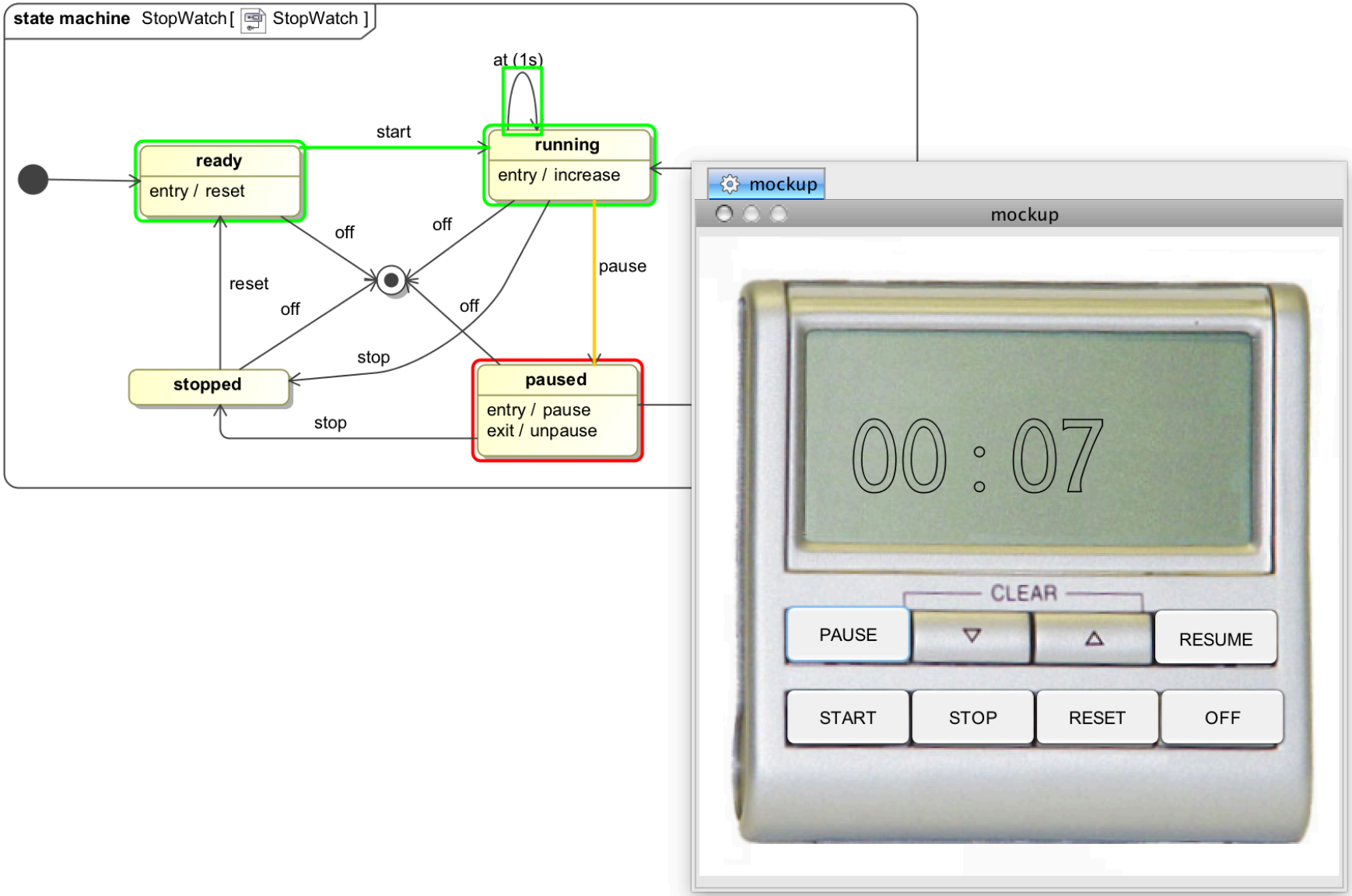
- Usage examples:

```
signal = ALH.getLastSignal($context$);
display = ALH.getValue($context$, "display");
ALH.setValue(car, ”speed", 60);
ALH.setValue($context$, "dot", false);
ALH.inState(self, “ready”)
```

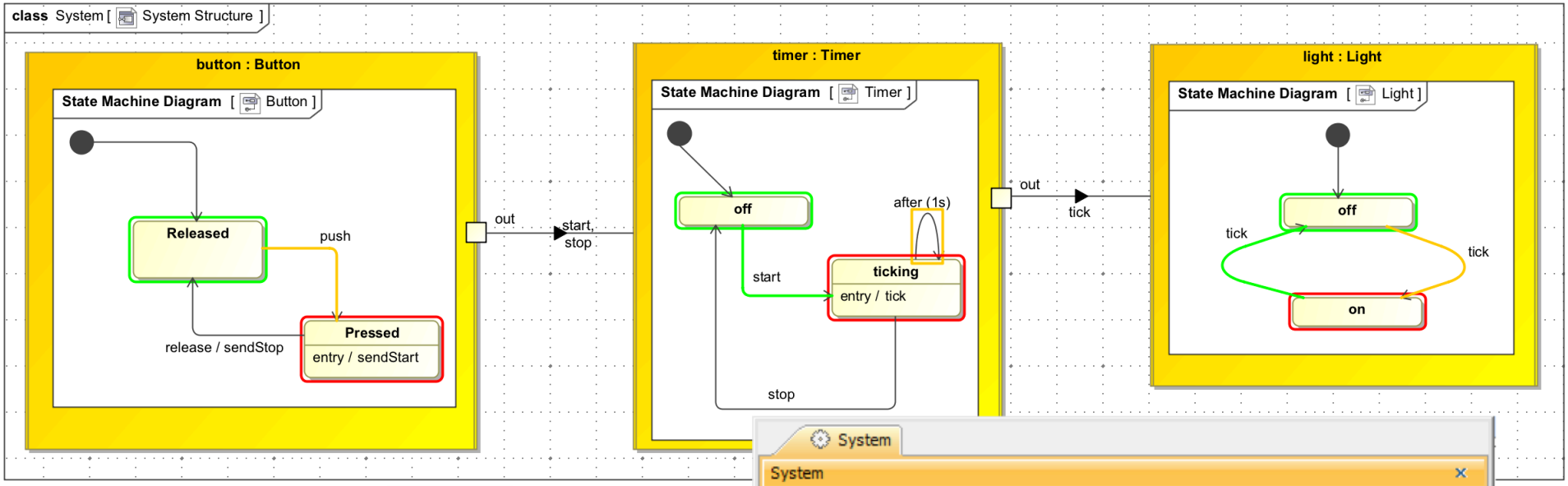
# Calculator model example

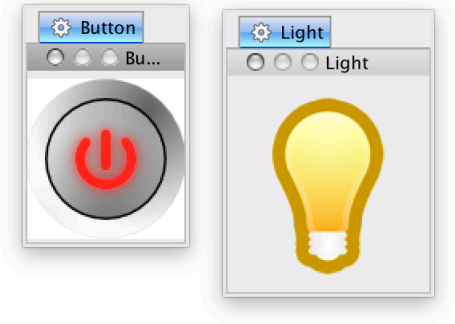


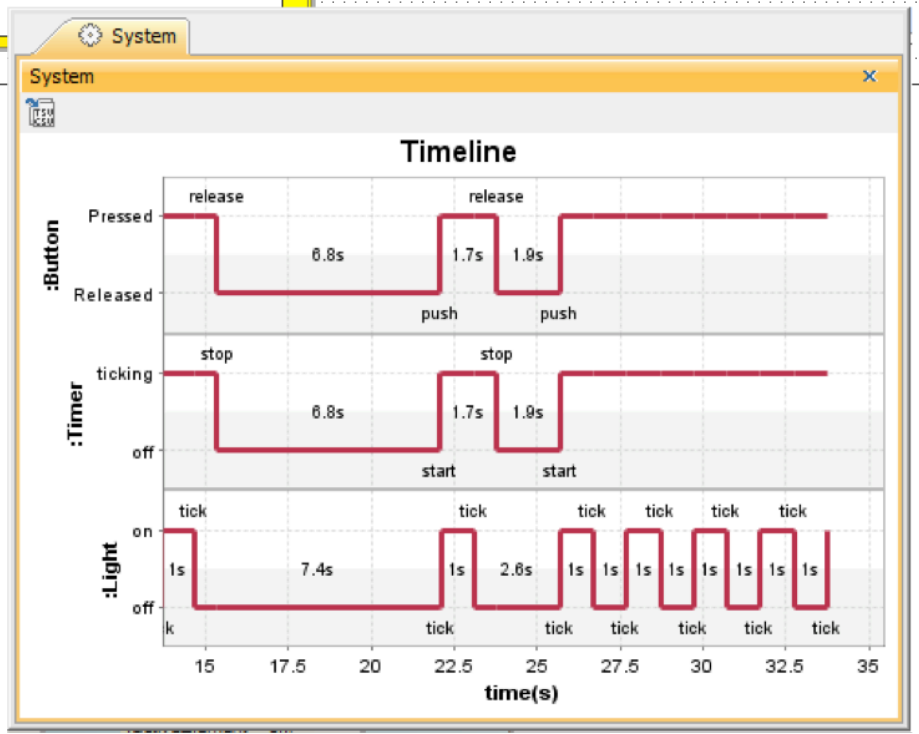
# Stopwatch example



# Flashing light example

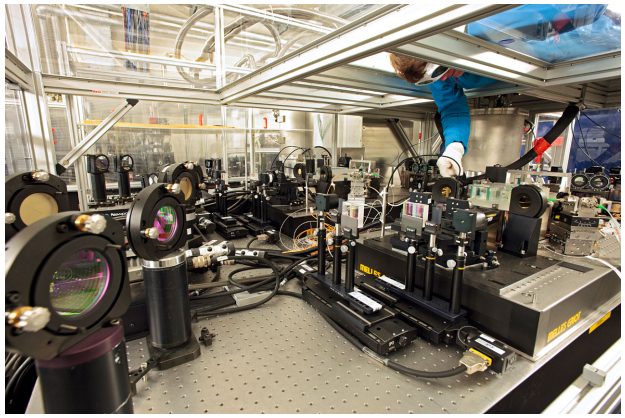


# ESO Applications

PRIMA



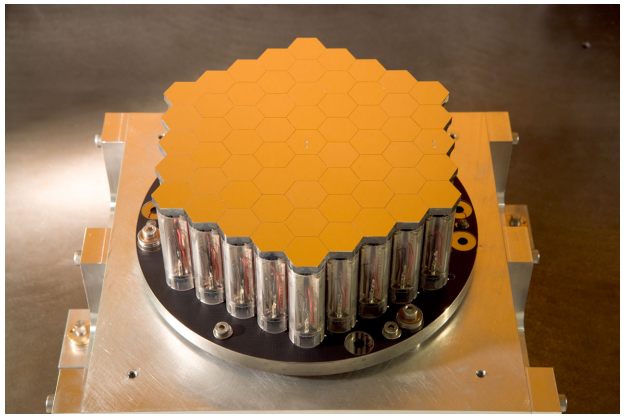
Auto-code generation

10 Components

States: 252

Transitions: 864

APE



Auto-code generation

11 Components

States:432

Transitions:1260

ELT

High level operations

17 Components

States: 34

Transitions:57

Activity elements: 340

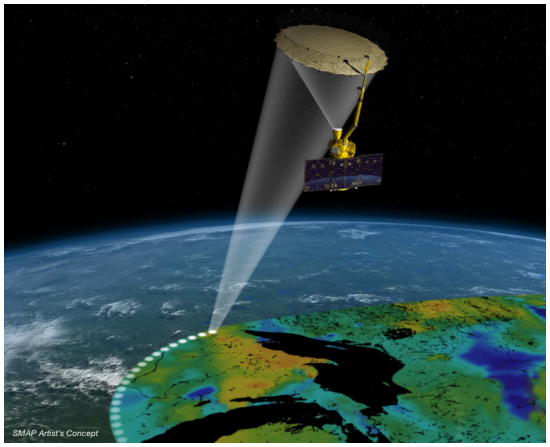
# NASA/JPL Applications

Thirty Meter Telescope APS

16 Components

States: 62
Transitions: 116
Signals: 100
Activity elements: 432
Duration constraints: 64

SMAP satellite



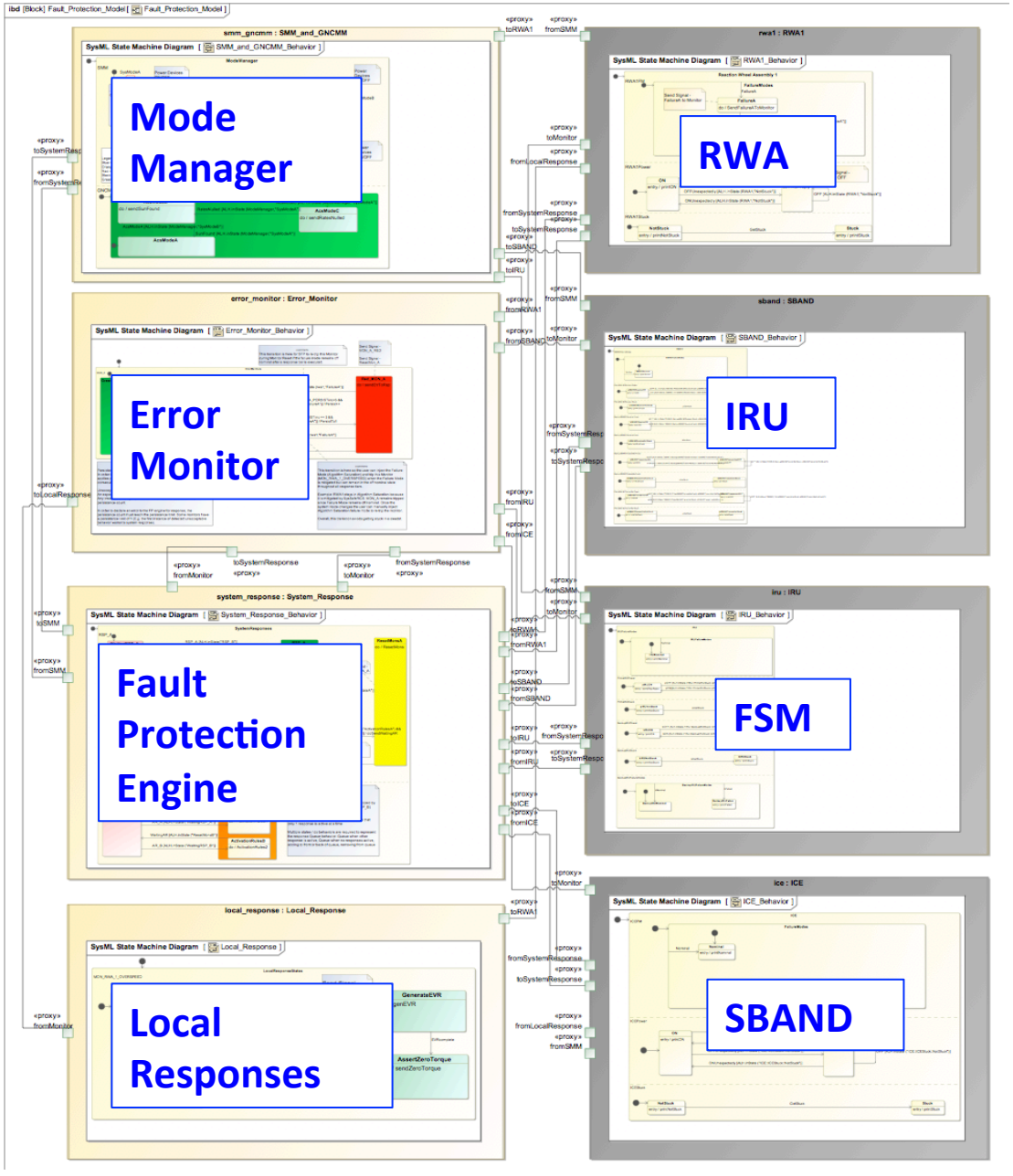

Modeling Fault Protection

40 Components

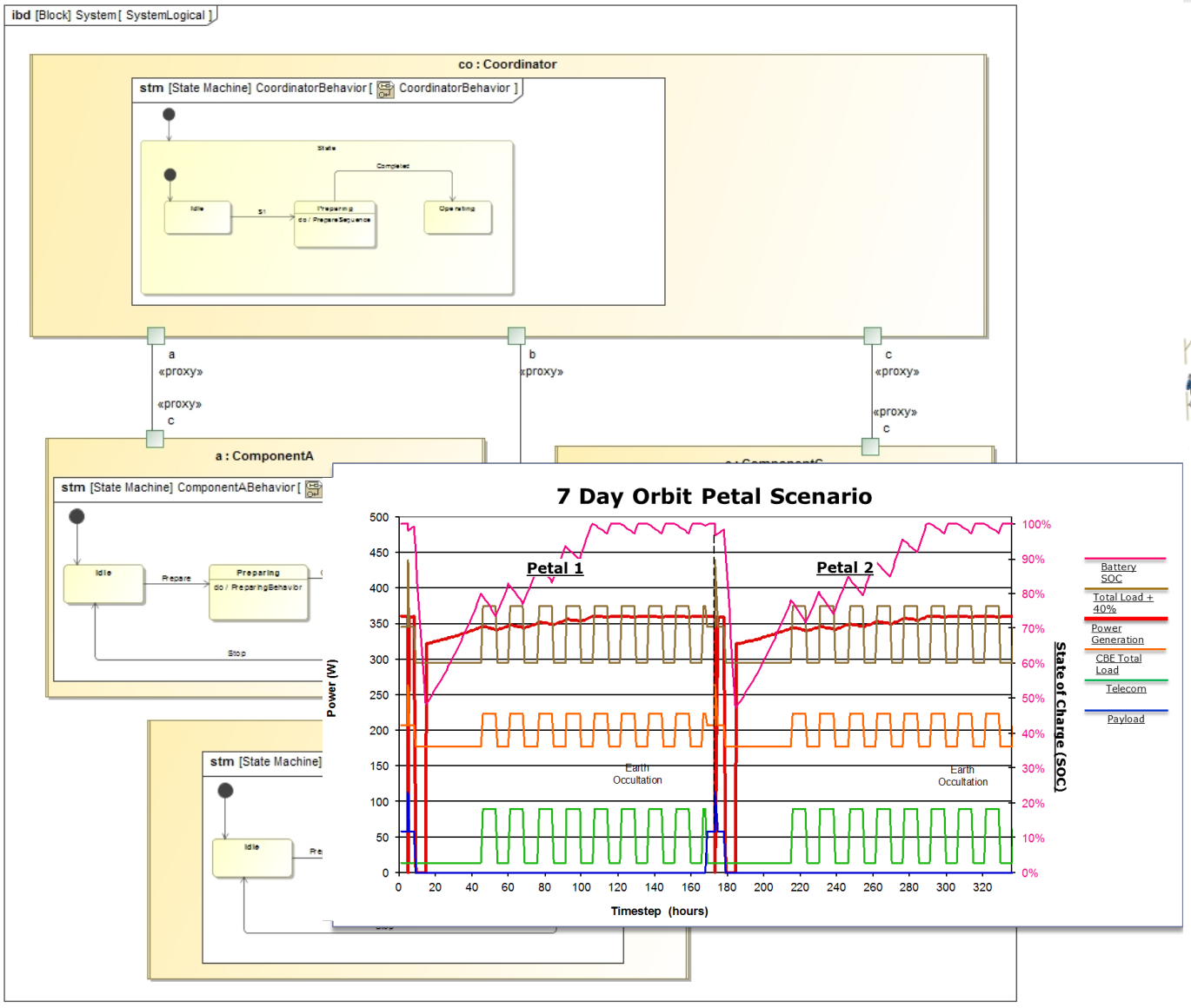
States: 2000
Transitions: 3000
Parallel regions: 600
Signals: 500

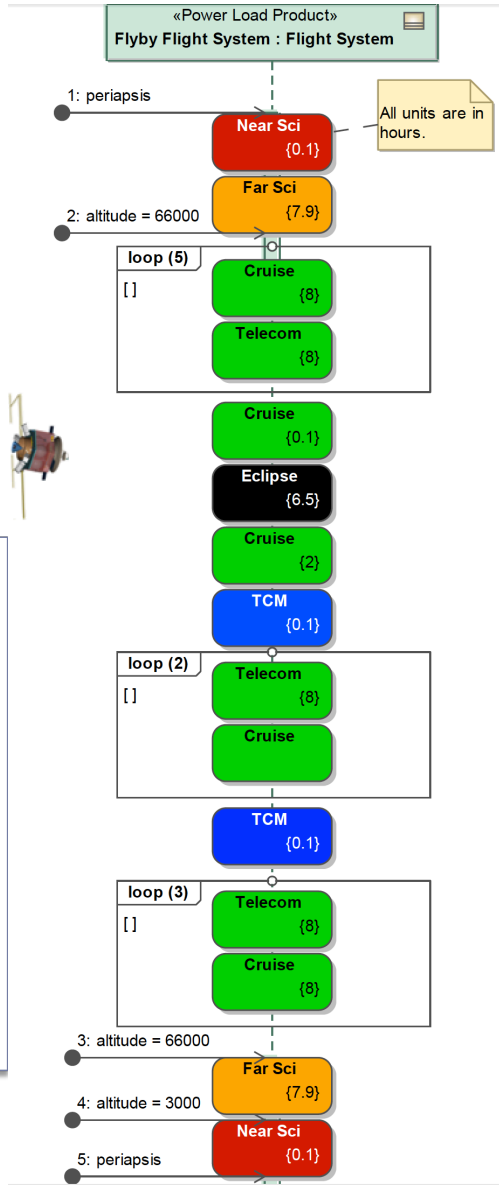
# Fault Protection Behavior

- Executable and Model-Checkable Statechart model
- Represents subset of Fault Protection logic for the JPL Soil Moisture Active Passive Satellite (launched Feb 2015)
- Basic SCXML-compliant statechart patterns captured even the most complicated Fault Protection logic



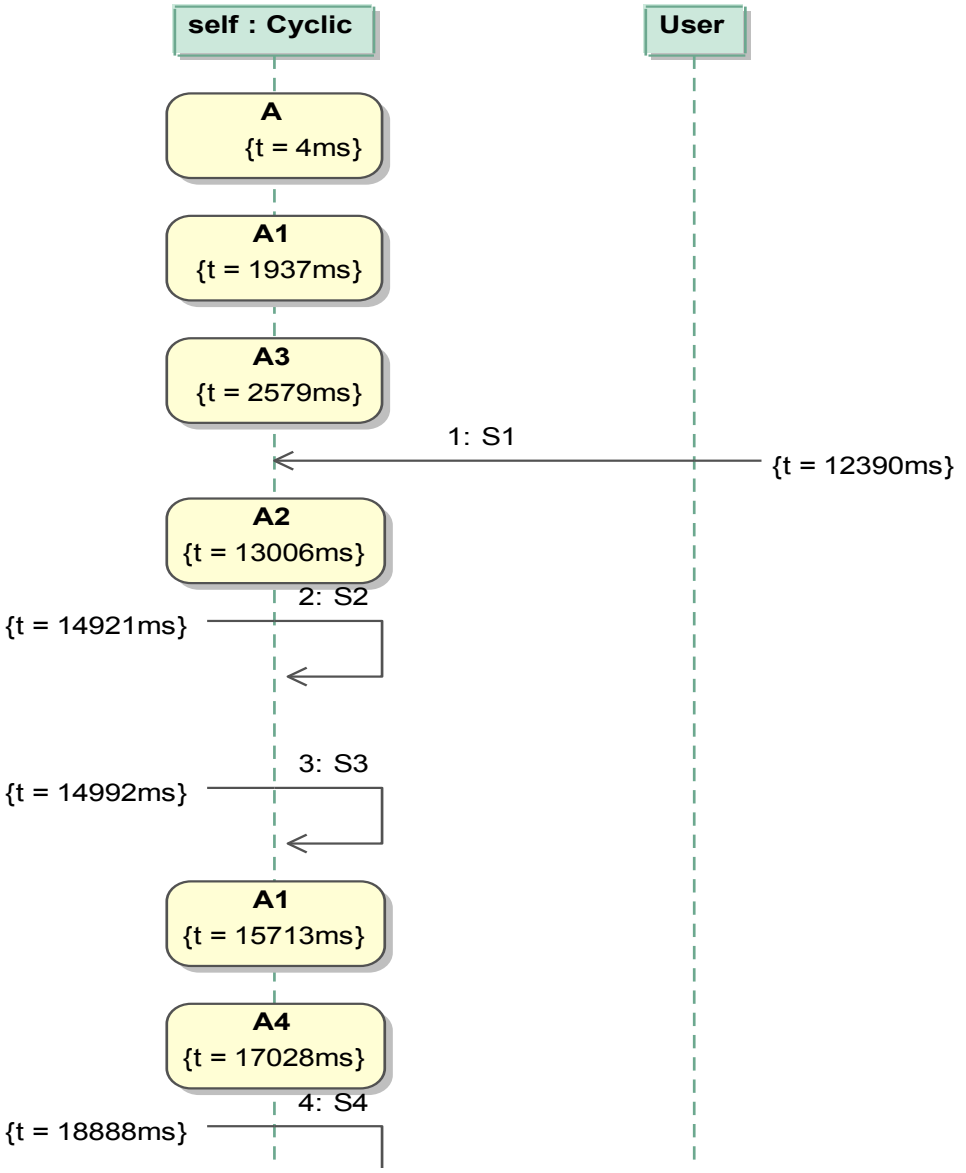
# Dynamic power rollup



# Generation of Sequence Diagram

- Execution tool outputs text log and visual sequence diagram for post-run behavior verification
- Goal is to run a handful of test cases for the different failure modes
- Verify that the correct responses that occur and that proper actions were taken to mitigate the fault
- Note: Not all cases can be simulated (takes too long) & interdependencies of multiple simultaneous fault injections can't be thoroughly simulated. – This is where model checking excels



# Timelines

- Time series chart
- Timing Diagram
- Sequence diagram recording
- CSV, TSV export
- Duration analysis