Extending Model-Based Systems Engineering with Requirements-In-the-Loop Simulation: The Landing Gears Case Study

Fabien GAUCHER

CATIA Systems Requirements R&D Director, Dassault Systèmes

CSD&M 2022
15-16 DECEMBER

CESAM COMMUNITY

AGENDA

- Simulating Requirements: Why?
- Overview of the landing gears case study
- Requirements-In-the-Loop Fundamentals
- Zoom on the physical and digital parts
- Integration with MBSE or MBD tools
- Conclusions and future work

Simulating requirements: Why?

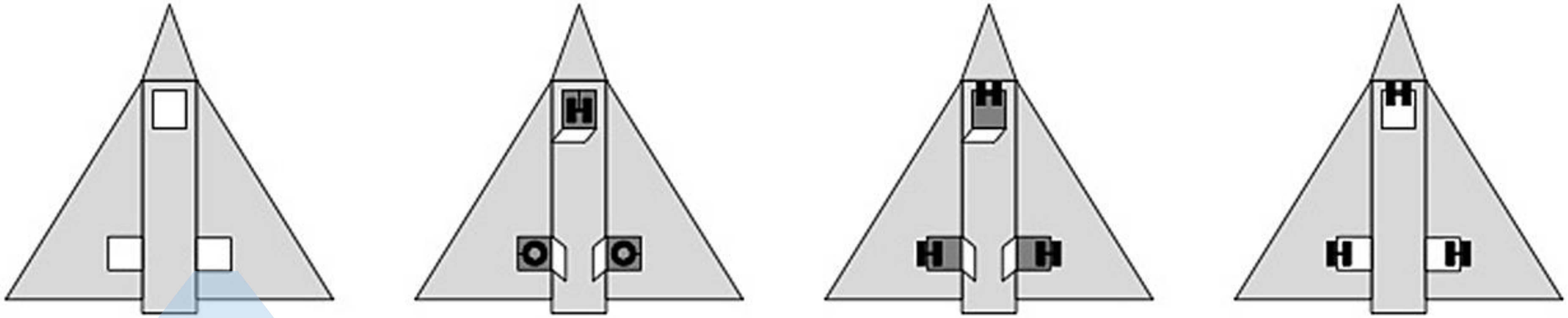
- ***Statement #1*** Over the last decades, simulation has been a key factor of the massive adoption by most industries of Model-Based Design and Model-Based System Engineering tools, as a way to tackle the ever-rising complexity of systems

- ***Statement #2*** In the meantime, industrial practice of Requirements Engineering has been limited to the life cycle management of document-centric specifications, leaving requirements aside from MBD and MBSE simulation tools

- ***Statement #3*** As a matter of fact, requirements simulation is a natural step towards early system validation, especially as formal and executable requirements can be integrated with MBSE and MBD simulation tools for validation purpose

The landing gears case-study

From the paper "*landing gear system*" by Frédéric Boniol and Virginie Wiels

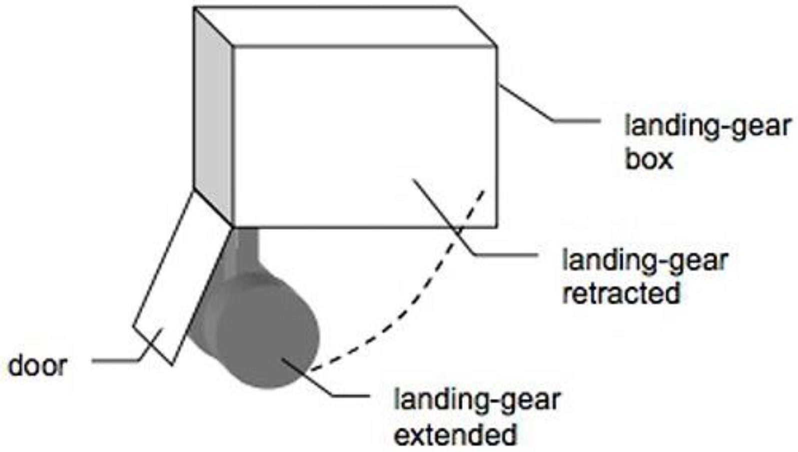


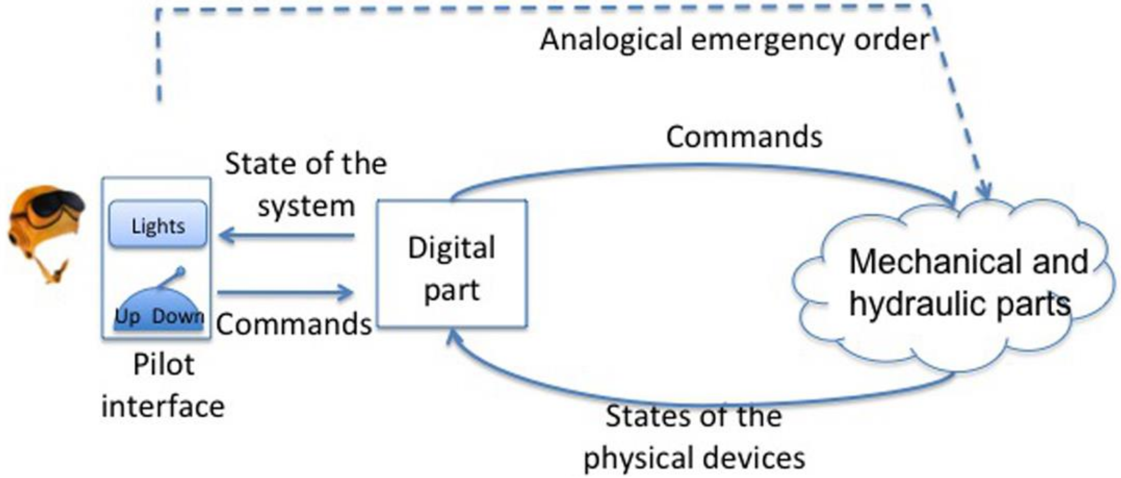
a) doors closed and gears retracted

b) doors opened and gears retracted

c) doors opened and gears extended

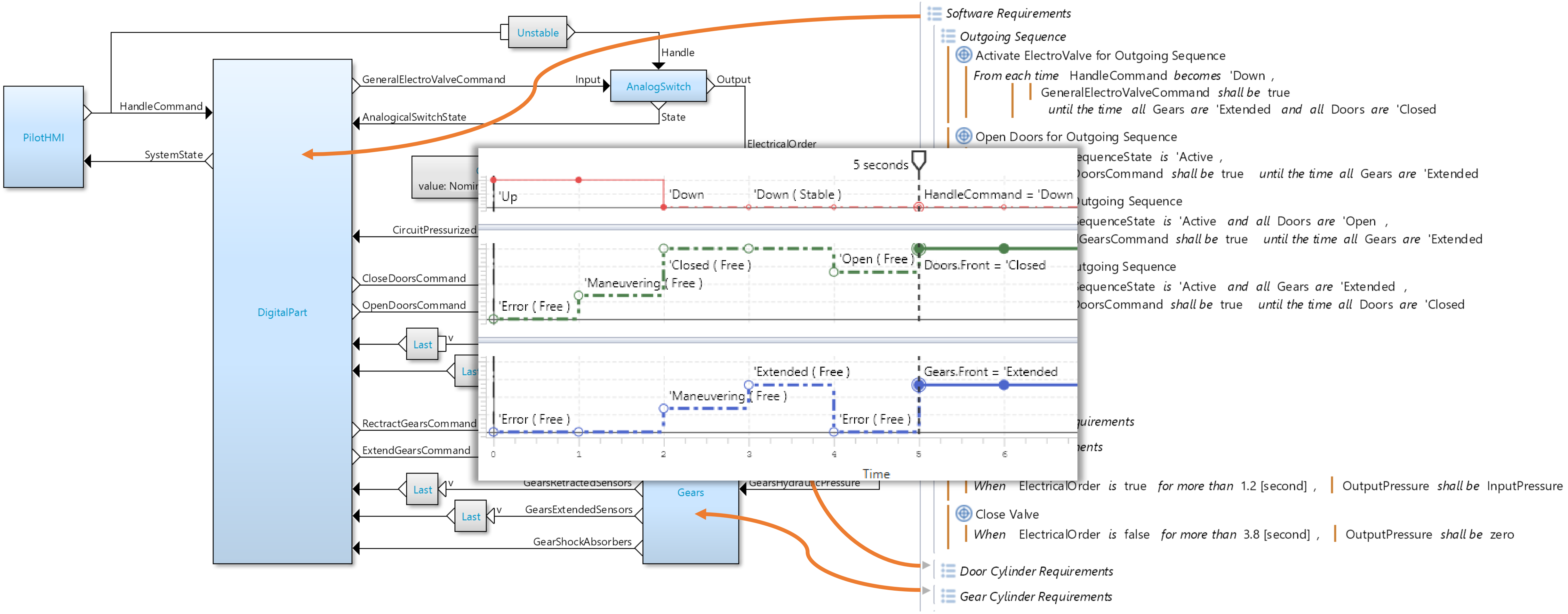
d) doors closed and gears extended





Requirements-in-the-loop Fundamentals

ALLOCATE FORMALIZED REQUIREMENTS TO SYSTEM ARCHITECTURE… AND SIMULATE!



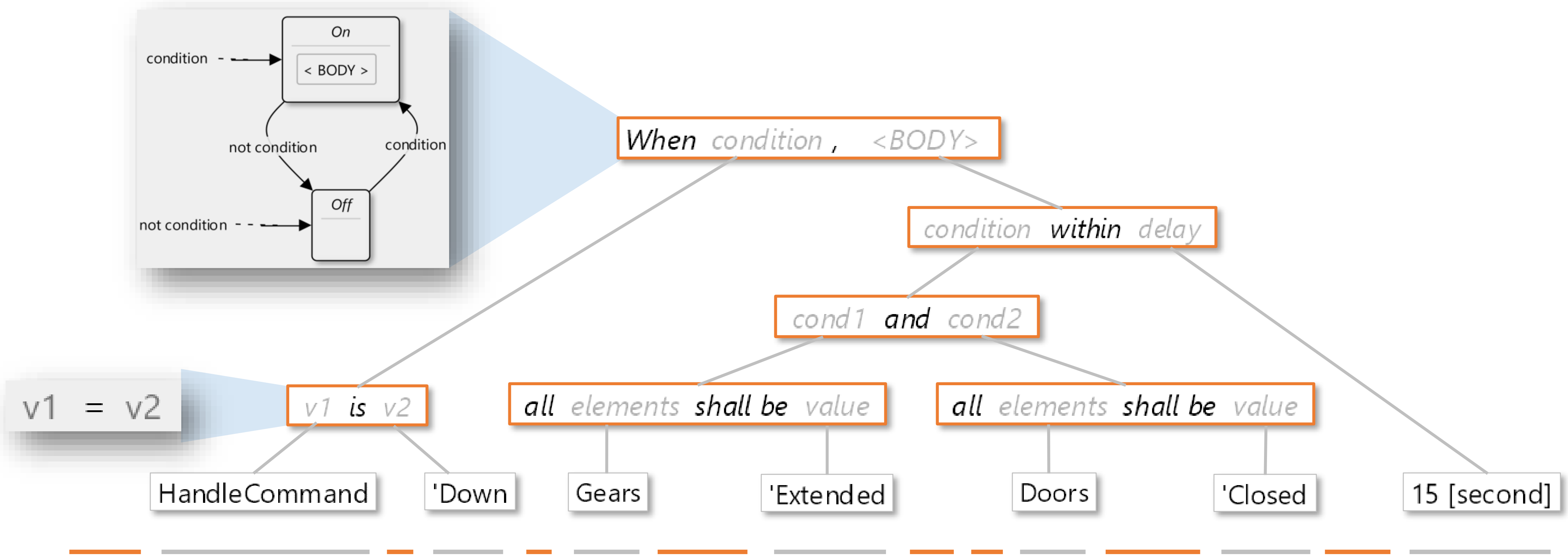
▶ System Architecture

▶ Formalized Requirements

CSD&M

Formalize textual requirements

WRITE UNAMBIGUOUS REQUIREMENTS BY COMPOSING TEMPLATES… THAT HAVE CLEAR EXECUTABLE SEMANTICS



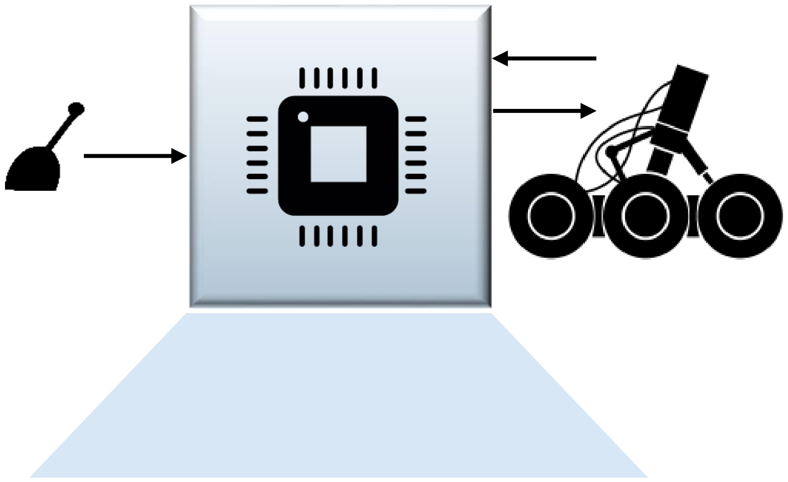
When HandleCommand *is* ‘Down*,* *all* Gears *shall be* ‘Extended *and* *all* Doors *shall be* ‘Closed *within* 15 [second]

▶ Resulting requirements can be simulated, but what does SIMULATION mean?

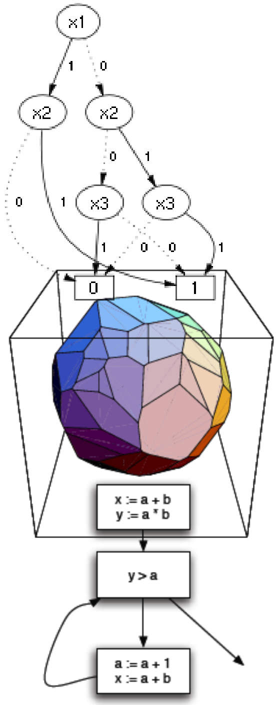
CSD&M

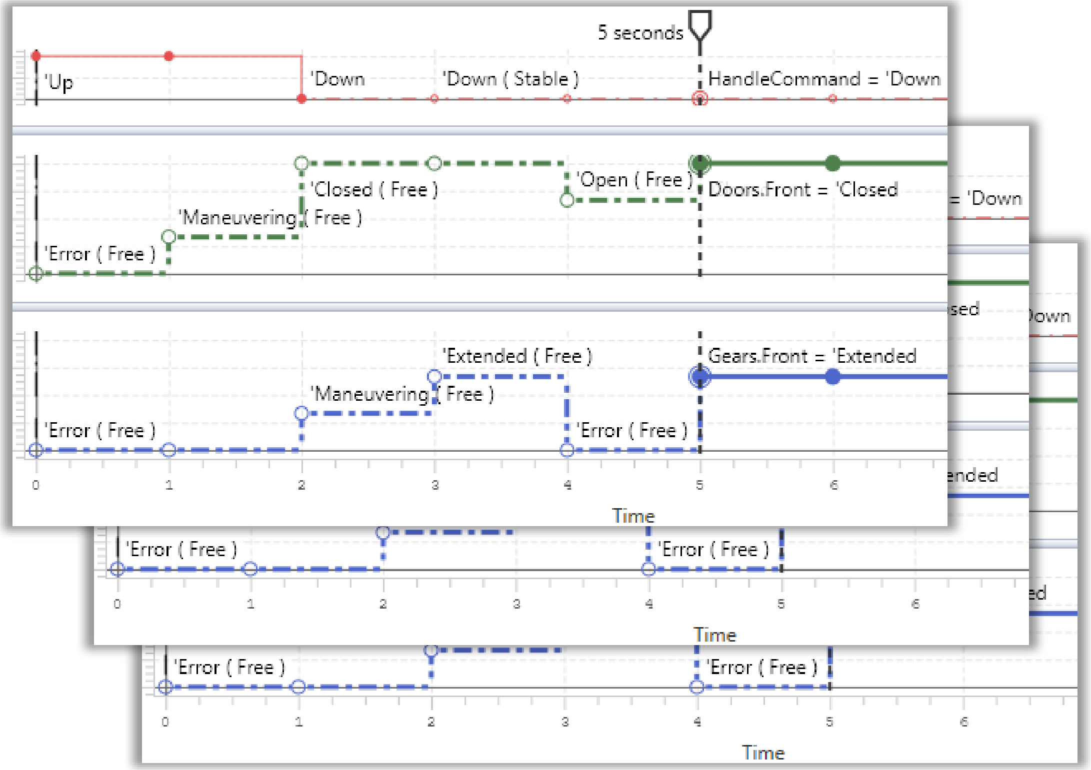
Simulate formalized requirements

GENERATE POSSIBLE BEHAVIORS THAT SATISFY REQUIREMENTS



When HandleCommand *is* 'Down,
all Gears *shall be* 'Extended *and*
all Doors *shall be* 'Closed
within 15 [second]





- Compiler transforms requirements into clocked equations
- Simulator solves logical, numerical and temporal constraints
- Debugger detects incorrect, missing or conflicting requirements

Specify the landing gear physical part

ZOOM ON THE ANALOGICAL SWITCH

▶ Original informal requirements

The switch is closed each time the handle is moved by the pilot, and *it remains closed for 20 seconds. After this duration, the switch automatically becomes open. In the closed position, the switch transmits the electrical order from the digital part to the general electro-valve. In the open position, no electrical order is sent to the electro-valve. Because of inertial reasons, the transition from the two states takes a given amount of time: from open to closed 0.8 second, from closed to open 1.2 seconds*

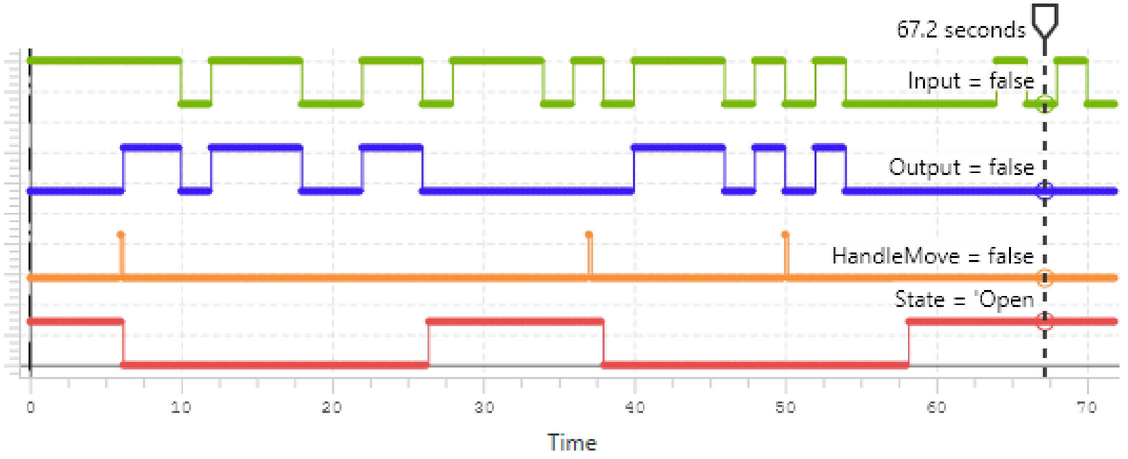
▶ CATIA Stimulus requirements

State Requirements

- Close switch on Handle
  - *After each time* HandleMove *becomes* true , | State *shall be* 'Closed *within* 0.8[second]
- Release switch after 20 seconds
  - 20 [second] *after the last time* State *became* 'Closed , | State *shall be* 'Open *within* 1.2[second]

Output Requirements

- Closed State
  - *When* State *is* 'Closed , | Output *shall be equal to* Input
- Open State
  - *When* State *is* 'Open , | Output *shall be equal to* false



- ▶ Each time the HandleMove signal is emitted, the switch is closed and the Input signal is transmitted to the Output signal.
- ▶ A deeper look reveals an incorrect behavior, as the HandleMove pulse at t=50s shall maintain the switch in the closed state until t=70s.
- ▶ Replacing the condition of the second requirement by “State became ‘Closed or HandleMove became true” is fixing the issue.

Specify the landing gears digital part

ZOOM ON THE GEARS OUTGOING SEQUENCE

▶ Original informal requirements

Outgoing sequence. The outgoing of gears is decomposed in a sequence of elementary actions. When the gears are locked in retracted position, and the doors are locked in closed position, if the pilot sets the handle to "Down", then the software should have the following sequence of actions:

1. stimulate the general electro-valve isolating the command unit in order to send hydraulic pressure to the maneuvering electro-valves,
2. stimulate the door opening electro-valve,
3. once the three doors are in the open position, stimulate the gear outgoing electro-valve,
4. once the three gears are locked down, stop the stimulation of the gear outgoing electro-valve,
5. stop the stimulation of the door opening electro-valve,
6. stimulate the door closure electro-valve,
7. once the three doors are locked in the closed position, stop the stimulation of the door closure electro-valve,
8. and finally stop stimulating the general electro-valve.

The previous sequences should be interruptible by counter orders (a retraction order occurs during the let down sequence and conversely) at any time. In that case, the scenario continues from the point where it was interrupted.

- ▶ Monolithic specification as related requirements cannot be considered independently
- ▶ Vague and non testable specification of counter-orders

▶ CATIA Stimulus requirements

Outgoing Sequence Requirements

Activate ElectroValve for Outgoing Sequence

When HandleCommand *is* 'Down ,
GeneralElectroValveCommand *shall be* true
until *all* Gears *are* 'Extended *and* *all* Doors *are* 'Closed

Open Doors for Outgoing Sequence

When HandleCommand *is* 'Down ,
OpenDoorsCommand *shall be* true *until* *all* Gears *are* 'Extended

Extend Gears for Outgoing Sequence

When HandleCommand *is* 'Down *and* *all* Doors *are* 'Open ,
ExtendGearsCommand *shall be* true *until* *all* Gears *are* 'Extended

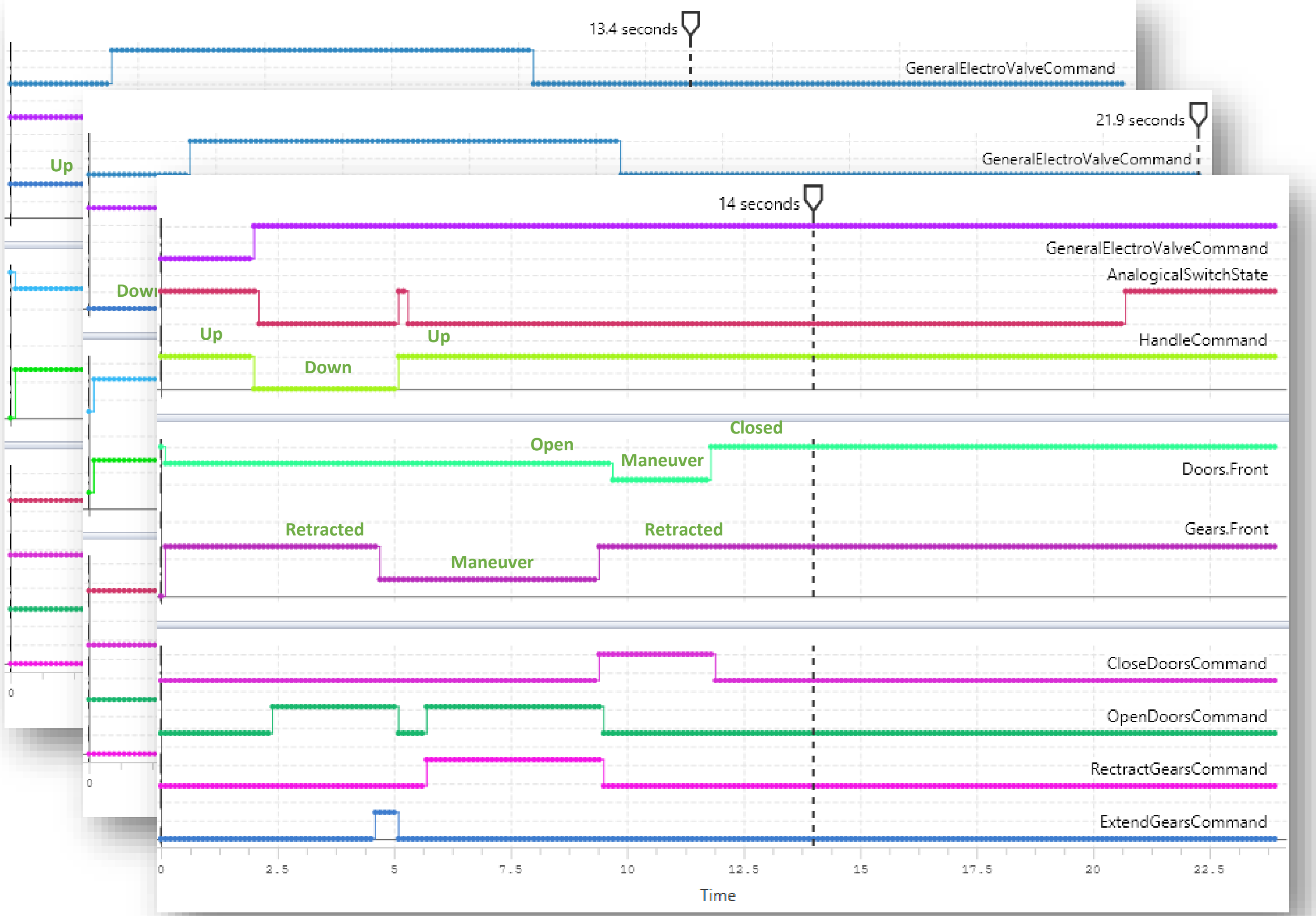
Close Doors for Outgoing Sequence

When HandleCommand *is* 'Down *and* *all* Gears *are* 'Extended ,
CloseDoorsCommand *shall be* true *until* *all* Doors *are* 'Closed

- ▶ Independent requirements which shall always be true whatever the state of the system
- ▶ Robust to counter-orders, testable
- ▶ Debugged through trials & errors thanks to simulation

Test the landing gears system

AUTOMATICALLY EXPLORE VARIOUS INITIAL CONDITIONS

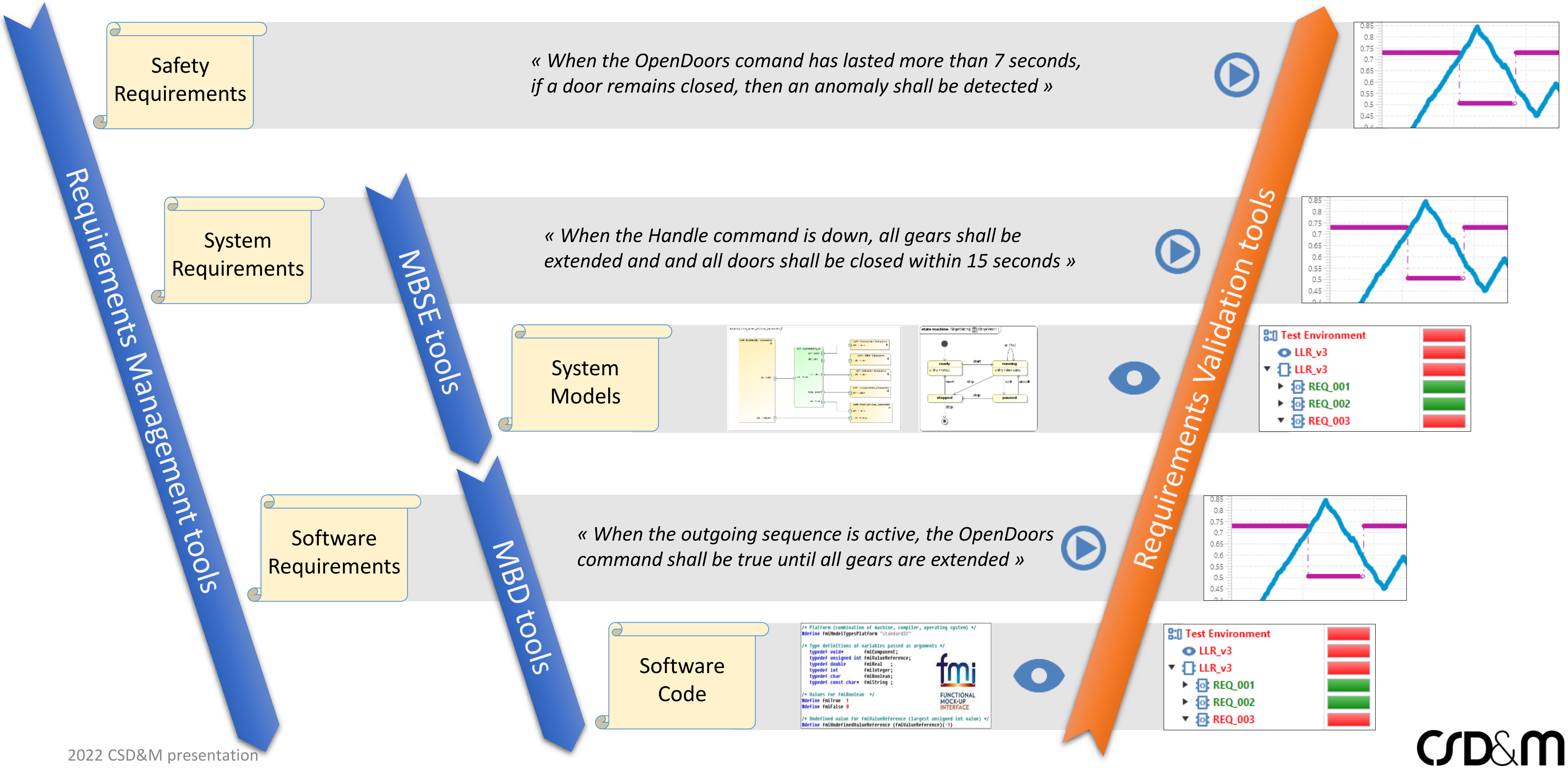


- When handle becomes down, gears go from the retracted to extended position, and doors from (already) open to closed
- When handle becomes up, gears don't move as they are already retracted, and doors go from open to closed position
- Counter-order during maneuvering is perfectly managed to reach the last command state (retracted)

CSD&M

Positioning with MBSE and MBD tools

END-TO-END USE OF RIL SIMULATION TO VALIDATE ARCHITECTURE AND BEHAVIORAL MODELS



Conclusions

- The landing gears case-study provides a typical embedded system specification with realistic complexity
- Requirements simulation is perfectly suited to debug and validate architectures and textual requirements
- Refinement of textual requirements into state machines or any FMU component can be addressed as well
- Future work includes configuration of simulation components and failure analysis

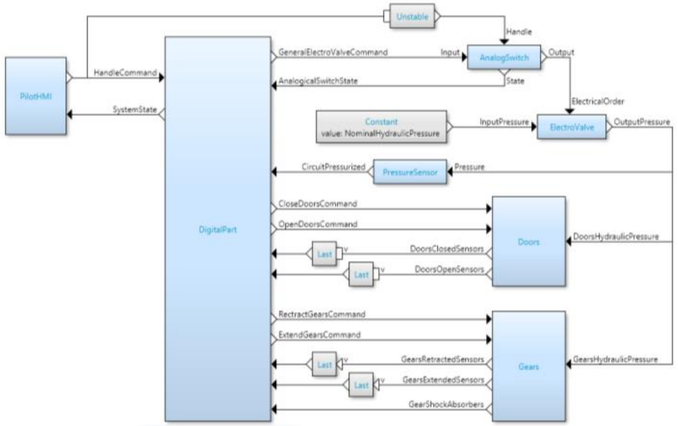
QUESTIONS?

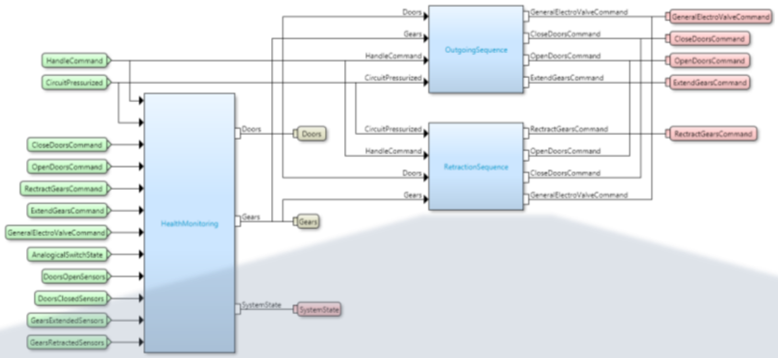
CATCH ME FOR LIVE DEMOS!

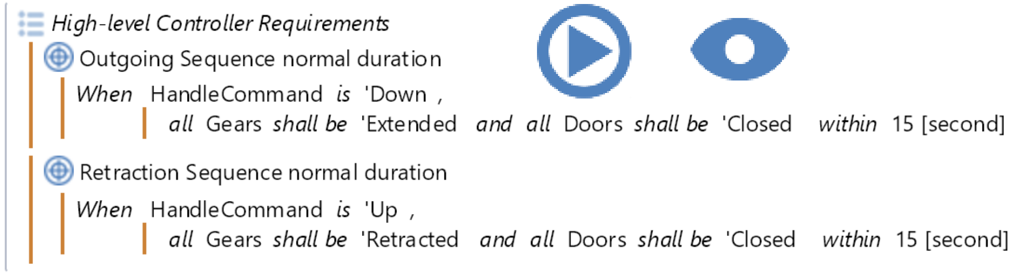
RIL Co-simulation

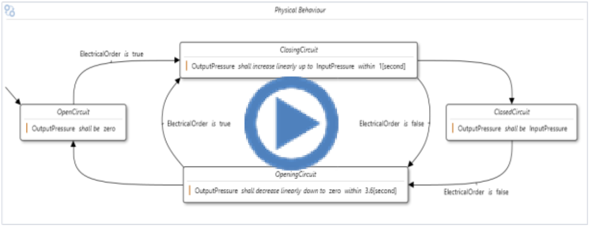
SINGLE ARCHITECTURE, MANY BEHAVIORS

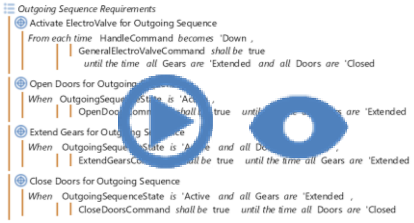
- A simulation configuration allows to select different behaviour models (block diagrams, state machines, external code or requirements) at each system level
- Requirements are automatically turned into observers when another behavior model is selected for simulation







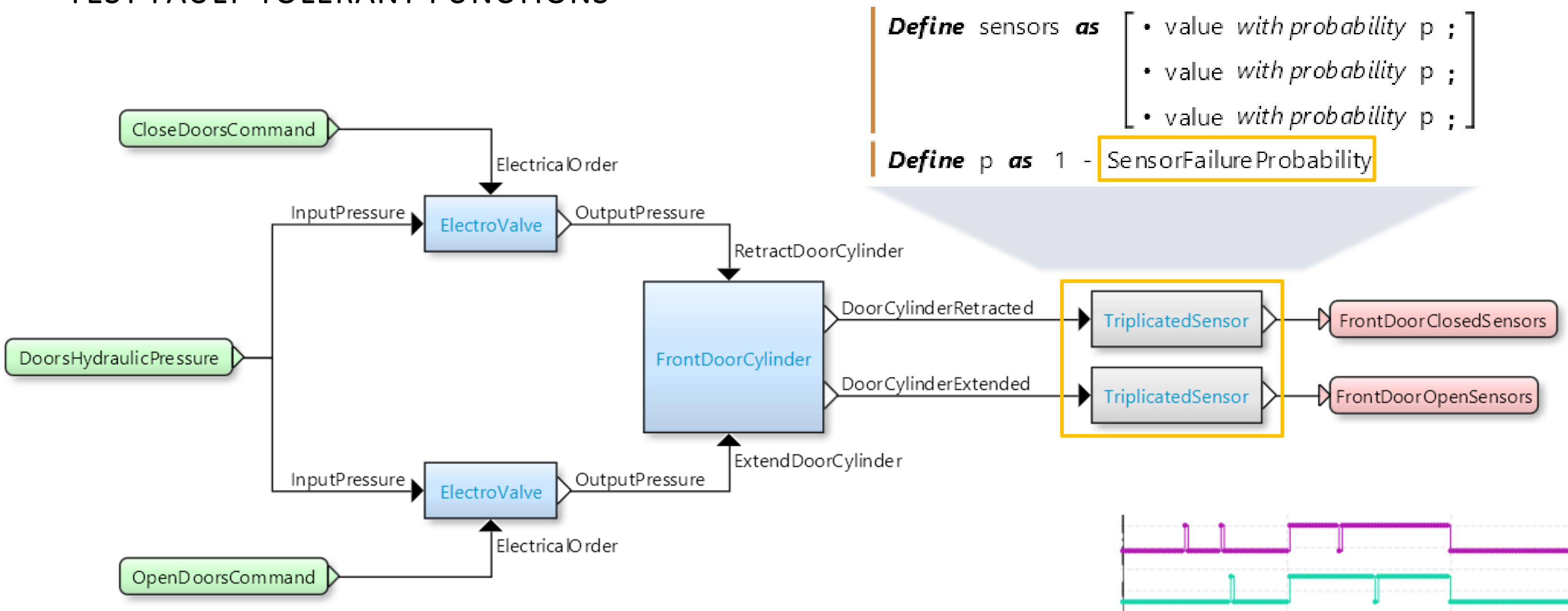


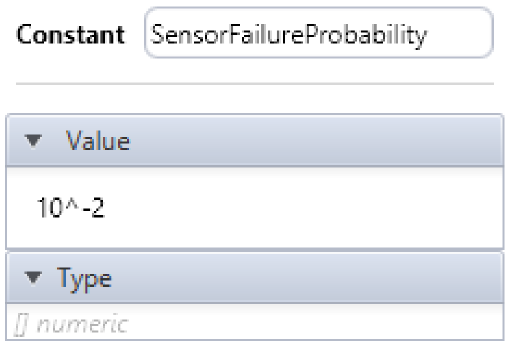


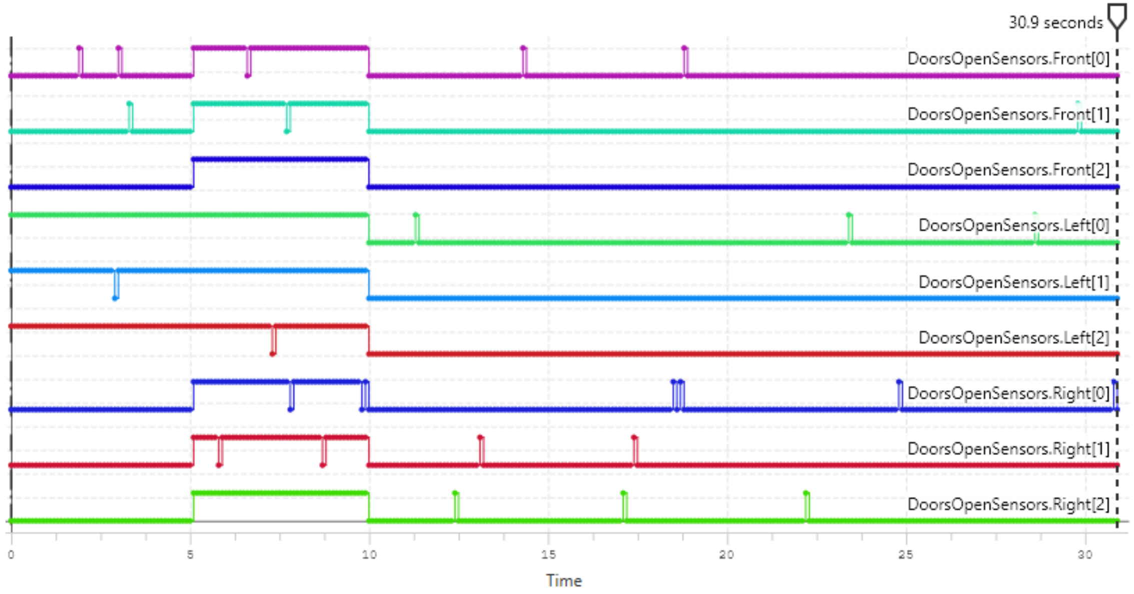
CSD&M

Failure analysis

TEST FAULT-TOLERANT FUNCTIONS







- All sensors are triplicated.
- A probability of failure is defined for the sensors.
- Each sensor has a probability to return a valid value.

- Test the Health Monitoring system which is in charge of:
  - detecting invalid sensors
  - selecting values from valid sensors

CSD&M

Contact

Fabien.Gaucher@3ds.com