Computational SysML Models

Bruce Powel Douglass, Ph.D.
Chief Evangelist, IBM IoT
Bruce.Douglass@us.ibm.com
www.bruce-douglass.com
Twitter: @IronmanBruce

Where in the Lifecycle is SysML Computational?

Engineering Analysis

Trade-off Analysis

"Analysis Time"

Model Verification

System Simulation

Digital Twin

System Execution

"Run Time"

A note on terms

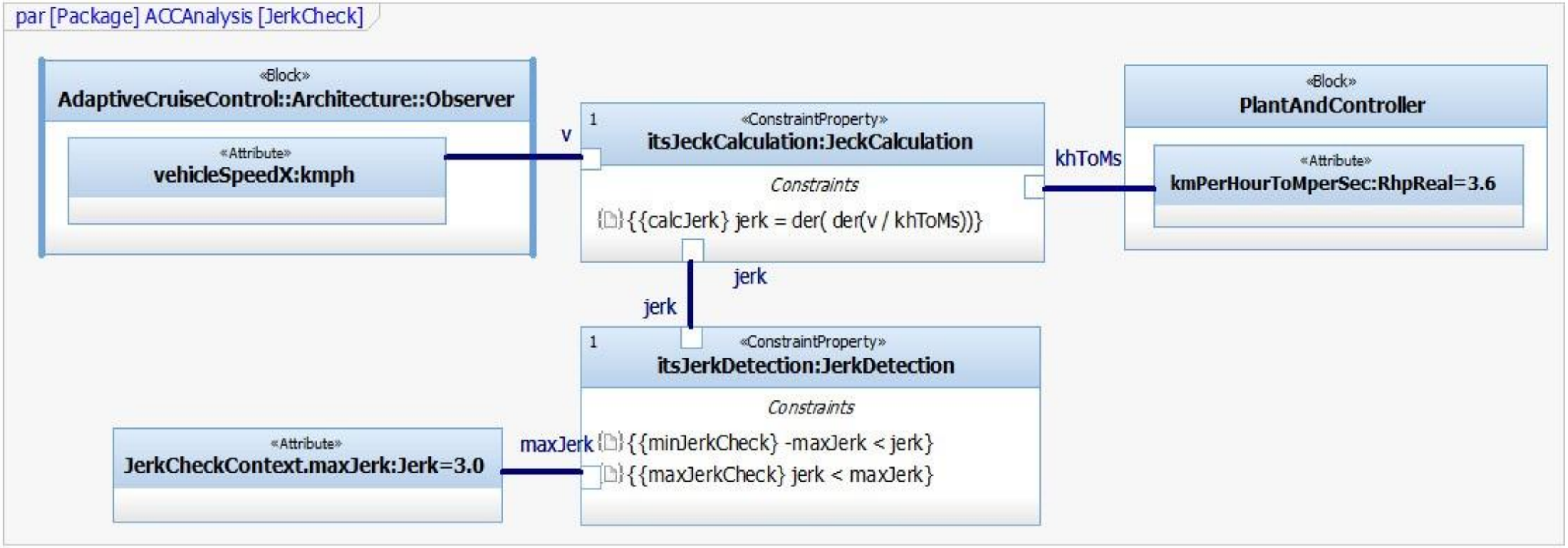
- A **computational** model is ultimately one that can be expressed mathematically in an *evaluable fashion*
  - An **executable** model is a computational model that is evaluated in a generated running system, whether as a simulation or an actual delivered system
  - An executable model is a “computational model with a direction of computation”
  - For example, f = m a
    - Computationally, if any two values are known, the third value can be computed. Such a model is **evaluable** by equation solvers.
    - However, if declare f to be the dependent variable, then it becomes **executable**.
- Computational models come in roughly two flavors, depending upon when the computation occurs.
  - Computational analysis models are evaluated at “analysis time” or “design time”
    - In SysML, this is normally specified with constraint properties on parametric diagrams. These can be evaluated by linking to computational engines such as MATLAB or Maxima
  - Computational design models are evaluated at “run time” either as simulations or actual delivered systems
    - In SysML, this is normally specified as state or activity diagrams, but may be augmented with methods outside of SysML, such as with FMI/FMU, Modelica, SimulationX, or Simulink

Computational Analysis Models

- Purpose: analyze proposed system properties to guide engineering decision making
- Examples
  - Determine system safety from analysis of fault probabilities
  - Determine optimal technology selection from alternatives (trade studies)
  - Analyze important system properties under conditions of interest

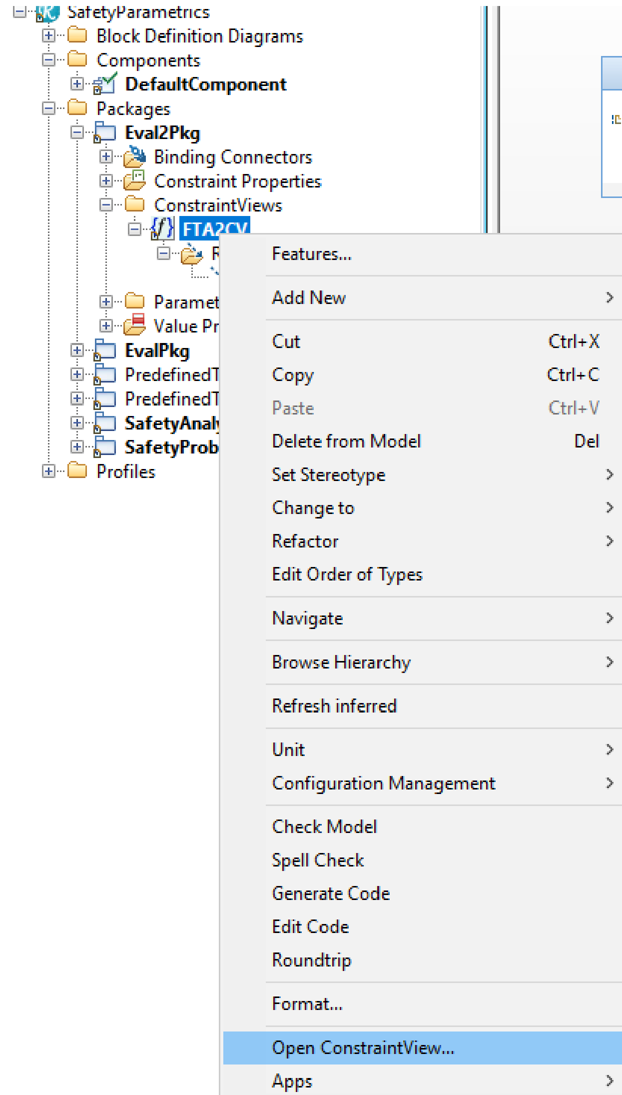
Analysis Time Computation: Parametric Diagram

- Imposes mathematical constraints on properties of Blocks (in system's context):
  - Constraint Block: groups non causal mathematical expressions (equations/inequalities)
  - Constraint Parameter: a variable of the math expressions that can be bounded to a design property
  - Constraint Property: a usage of a constraint block in a specific context
  - Binding Connector: declared that the value of the design property must be equal to the value of the constraint parameter



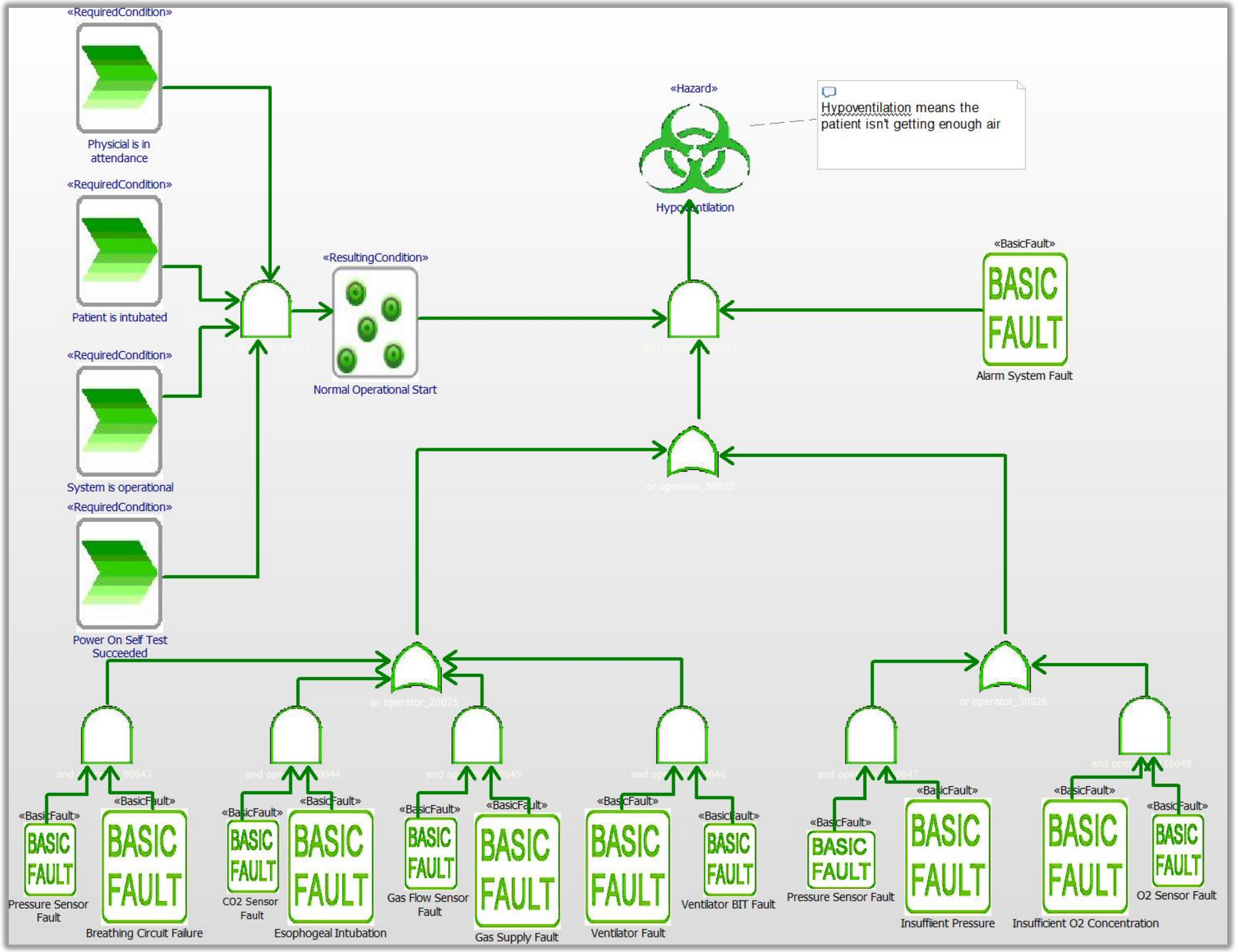
Using Parametric Constraint Evaluator Profile

- UML/SysML tools are not generally capable of computational analysis. However, they can capture constraints in such a way that they can invoke such tools to perform such analysis.
  - Example: SPT and MARTE profiles provide a standard means for specifying performance properties for schedulability analysis so that other tools – such as TriPacific's RapidRMA tool – can extract the information and "do the math."
  - Example: Rhapsody's Dependability Profile (available at www.bruce-douglass.com) allows you to specify the probability of fault occurrence but does not directly compute the probability of the resulting hazard.
- These problems can be expressed on SysML Parametric Diagrams but cannot be evaluated directly in SysML.
- Rhapsody provides a Parametric Constraint Evaluation (PCE) profile that allows you to link parametric diagrams (and contained constraint models) to either Matlab or Maxima for mathematical evaluation.



Example Fault Tree Analysis

- Each of the Fault and events have a likelihood (probability) or occurrence.
- Therefore, it is possible to compute the likelihood of the hazard using the connective logical operators AND, OR, NOT, NOR, and so on.

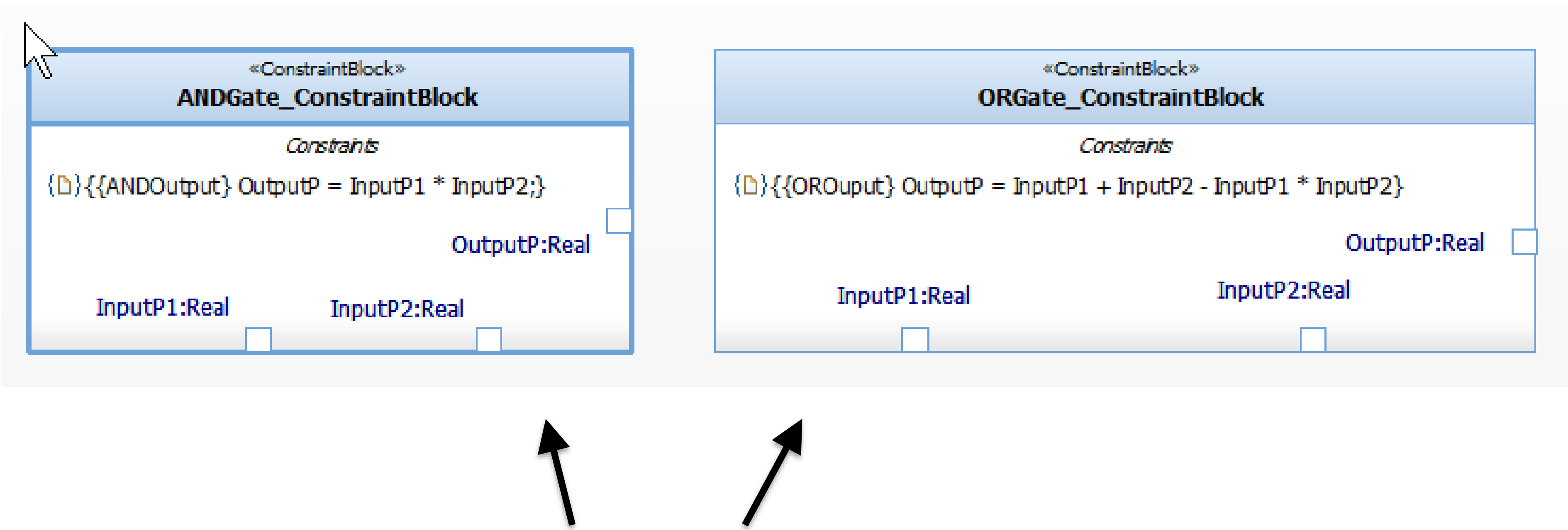


Calculating the likelihood of hazards

- You can calculate the hazard probability via "propagation of probabilities" by performing computations up the causal chain.
- Probability Computation
  - Step 1: Create FTA
  - Step 2: Document primitive fault probabilities (0.0 to 1.0)
    - Assume **Required Conditions** and **Required Events** have probability 1.0
  - Step 3: Write the FTA as a succession of equations
    - AND: $P_{AND} = P_1 * P_2$ where P_1 is the probability of input 1 & P_2 is the probability of input 2
    - OR: $P_{OR} = P_1 + P_2 – P_1 * P_2$
    - NOT: $P_{NOT} = 1.0 - P_1$
    - NAND: $P_{NAND} = 1.0 - P_1 * P_2$
    - NOR: $P_{NOR} = 1.0 - P_1 + P_2 – P_1 * P_2$
    - XOR: Remember: $P_{XOR} = (P_1 \text{ AND } (\text{NOT } P_2)) \text{ OR } ((\text{NOT } P_1) \text{ AND } P_2)$
      so $P_{XOR} = (P_1 * (1.0-P_2)) + ((1.0-P_1) * P_2) - (P_1 * (1.0-P_2)) * ((1.0-P_1) * P_2)$
  - Step 4: Do the math
  - Step 5: Repeat in the next step of the causal chain

Evaluating with a Parametric Diagram

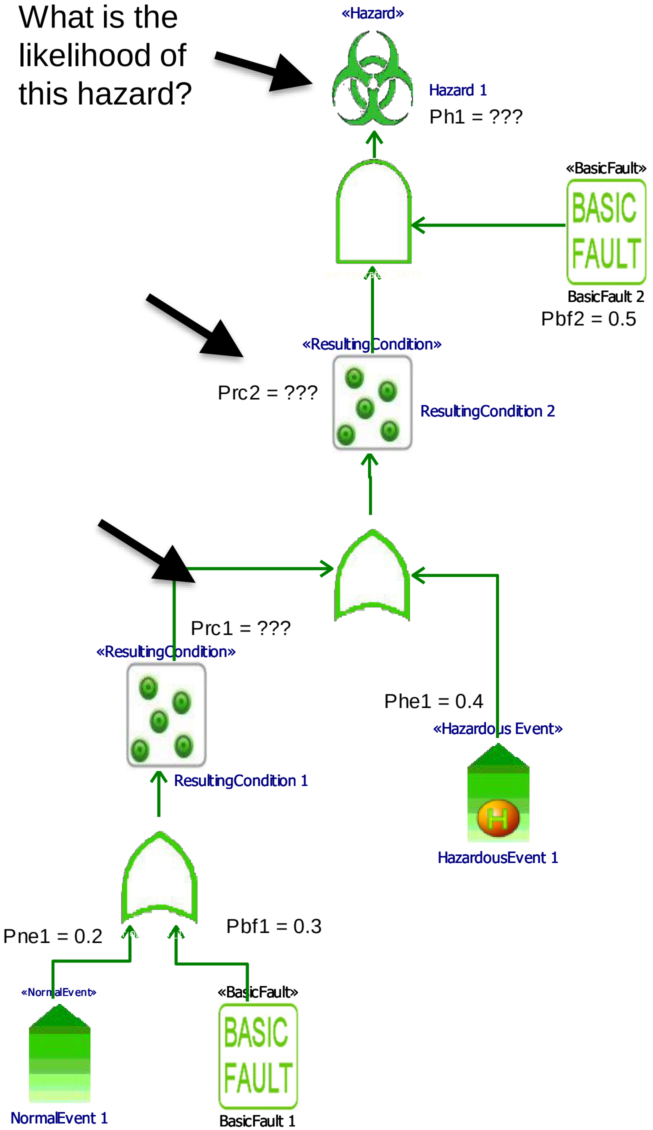
- Build a library of constraint blocks for the various gates:

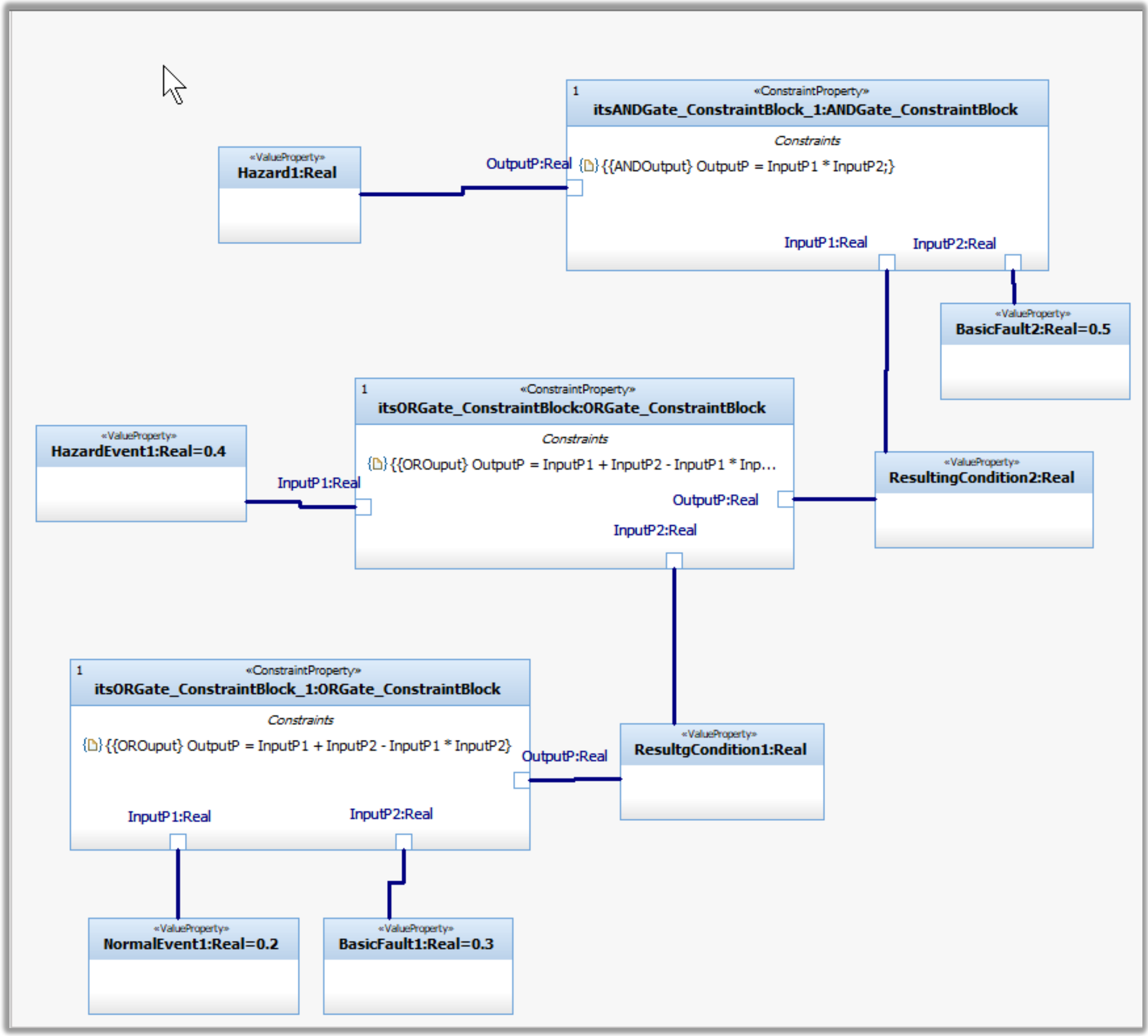


Constraint Blocks for the logic gates

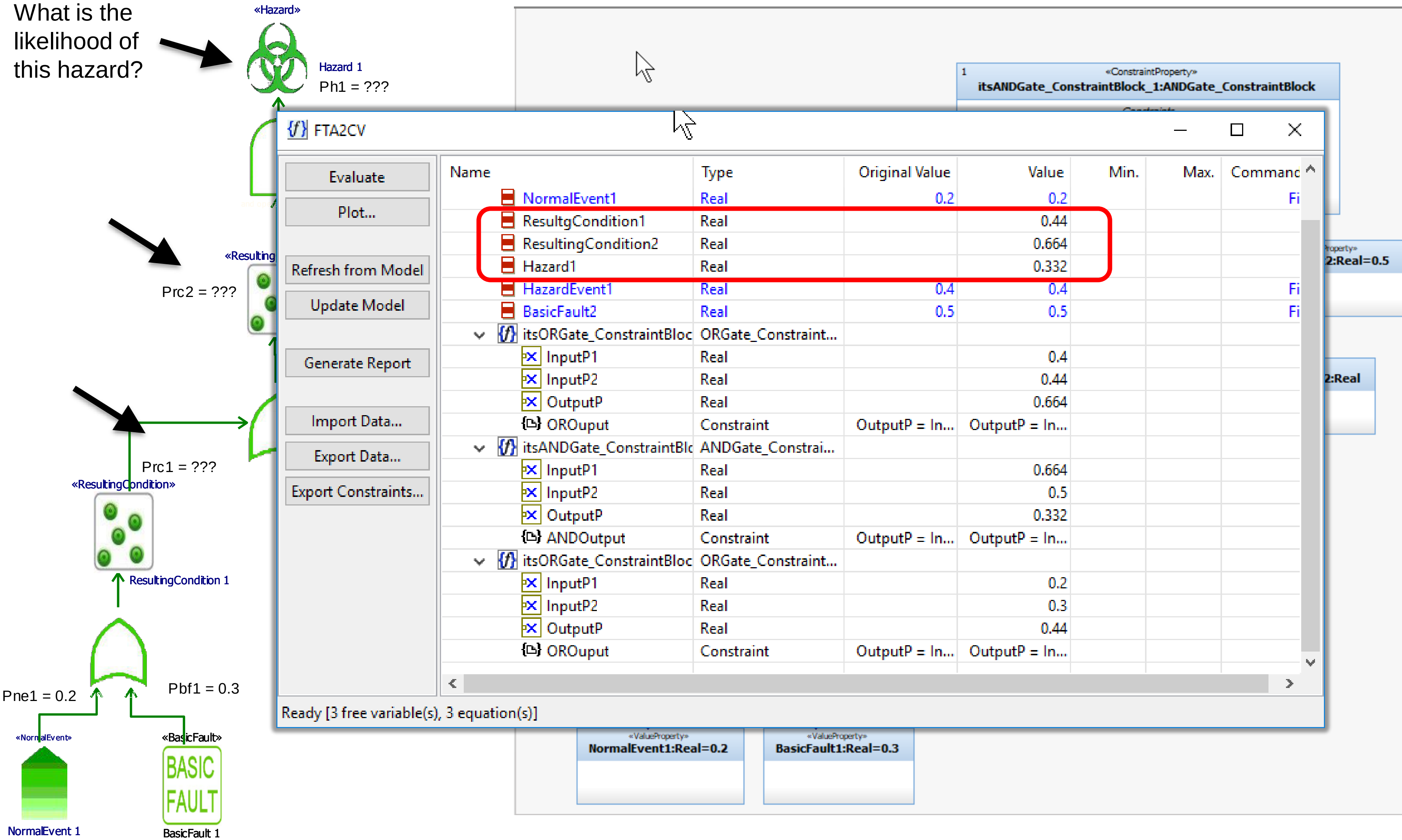
Calculating the likelihood of hazards: Doing the math

What is the likelihood of this hazard?





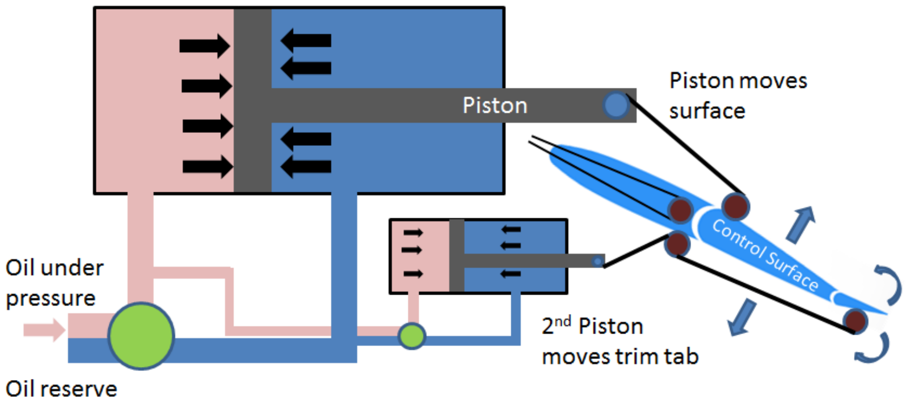
Calculating the likelihood of hazards: Doing the math



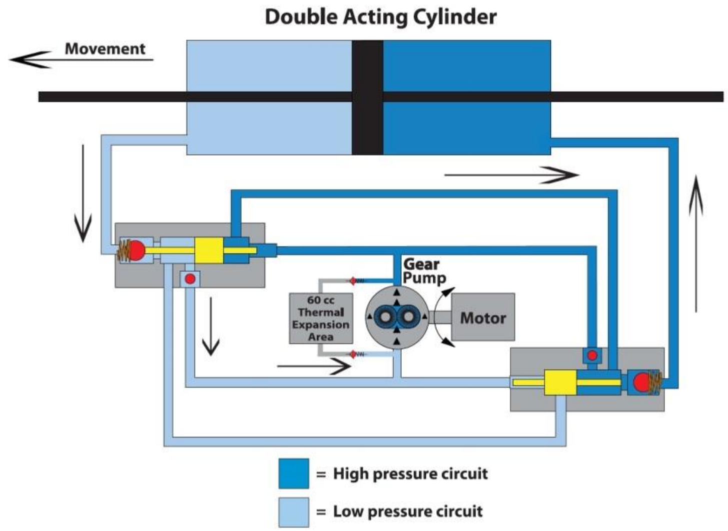
Name	Type	Original Value	Value	Min.	Max.	Command
NormalEvent1	Real	0.2	0.2			Fi
ResultgCondition1	Real		0.44			
ResultingCondition2	Real		0.664			
Hazard1	Real		0.332			
HazardEvent1	Real	0.4	0.4			Fi
BasicFault2	Real	0.5	0.5			Fi
itsORGate_ConstraintBloc	ORGate_Constraint...					
InputP1	Real		0.4			
InputP2	Real		0.44			
OutputP	Real		0.664			
OROuput	Constraint	OutputP = In...	OutputP = In...			
itsANDGate_ConstraintBlc	ANDGate_Constrai...					
InputP1	Real		0.664			
InputP2	Real		0.5			
OutputP	Real		0.332			
ANDOutput	Constraint	OutputP = In...	OutputP = In...			
itsORGate_ConstraintBloc	ORGate_Constraint...					
InputP1	Real		0.2			
InputP2	Real		0.3			
OutputP	Real		0.44			
OROuput	Constraint	OutputP = In...	OutputP = In...			

Architectural Trade Study Analysis

We will examine the trade offs for movement of the trim tabs and extension of some of the control surfaces, looking at three technical solutions:

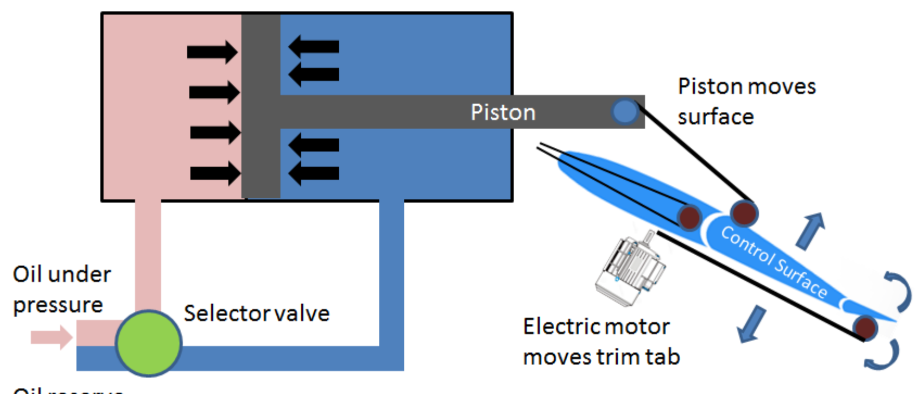


Hydraulic actuator



Electro-hydraulic actuator

From the **Harmony aMBSE Deskbook** available at https://www.bruce-douglass.com/papers

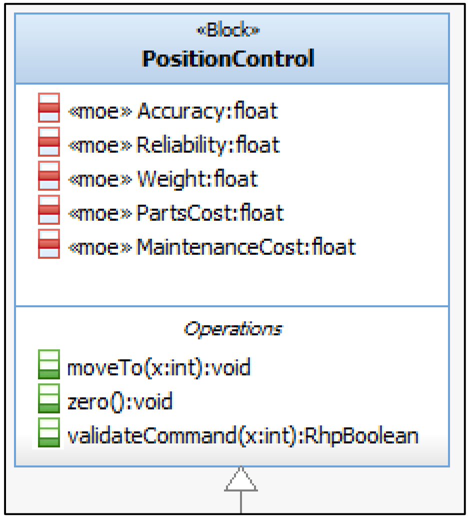


Electric motor

Architectural Analysis: Define Assessment Criteria

☑ Identify the assessment criteria:
- Accuracy of movement
- Weight
- Reliability
- Parts cost
- Maintenance cost

- Assign them normalize weight (importance) values
  - Accuracy of movement 0.30
  - Weight 0.20
  - Reliability 0.25
  - Parts cost 0.10
  - Maintenance cost 0.15



Architectual Analysis: Define the Utility Curves

☑ Obtain the values of the MOEs for all the technical solutions

Solution/ moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100

☑ Define the (linear) utility curves so that the worst solution returns a value of 0 and the best solution returns a value of 10

$$accuracyMOE = -\frac{5}{2}accuracy + \frac{25}{2}$$

$$weightMOE = -\frac{5}{24}weight + 15$$

$$reliabilityMOE = \frac{reliability}{80} - 40$$

$$partCostMOE = -\frac{partsCost}{25} + 32$$

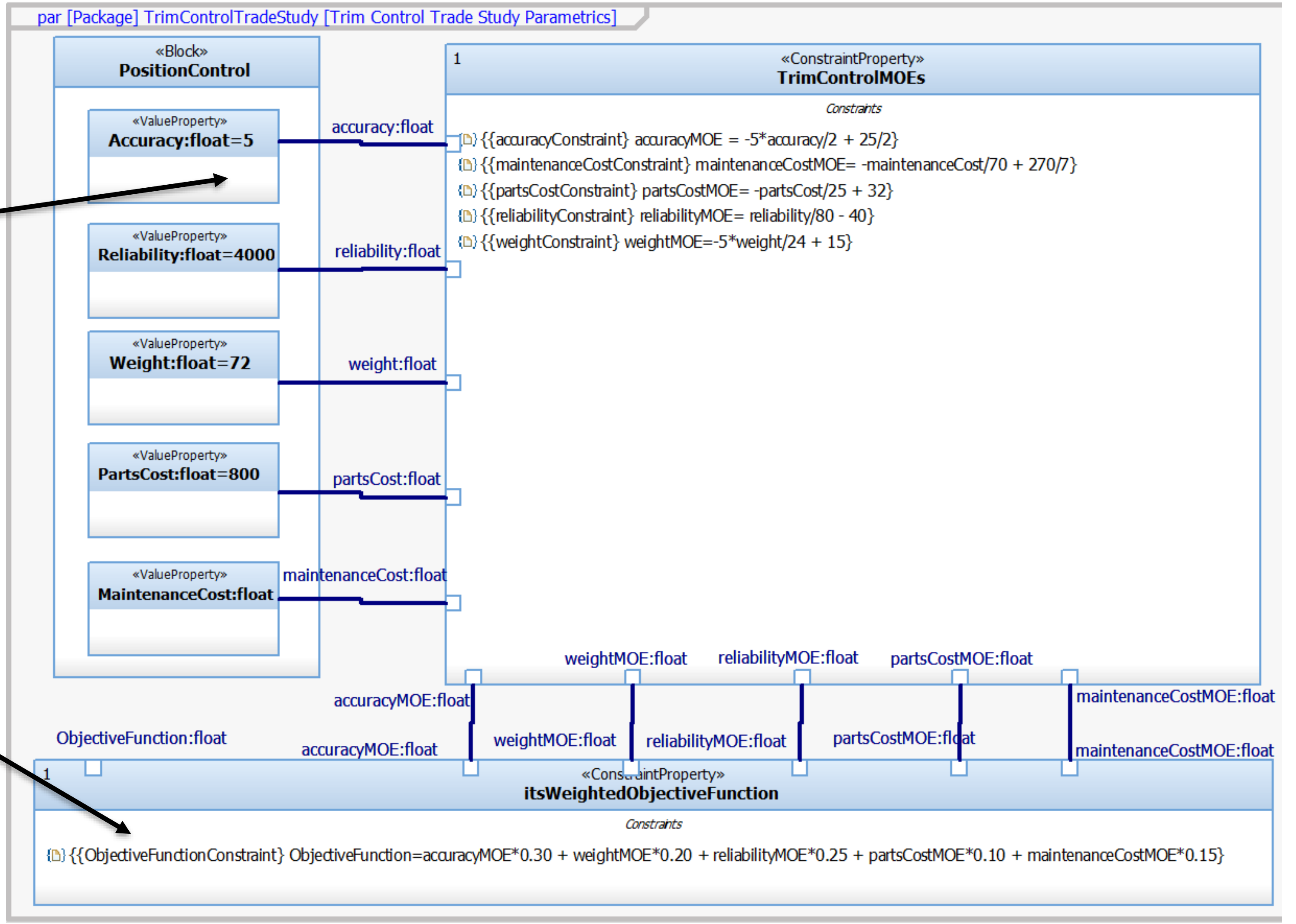
$$maintenanceCostMOE = -\frac{maintenanceCost}{70} + \frac{270}{7}$$

Architectural Analysis: Define Assessment Criteria

☑ Capture the utility functions on a parametric diagram

Note: To evaluate, the "initial value" of each of the value properties must be set, and then the constraint blocks are evaluated for the specific set of values.

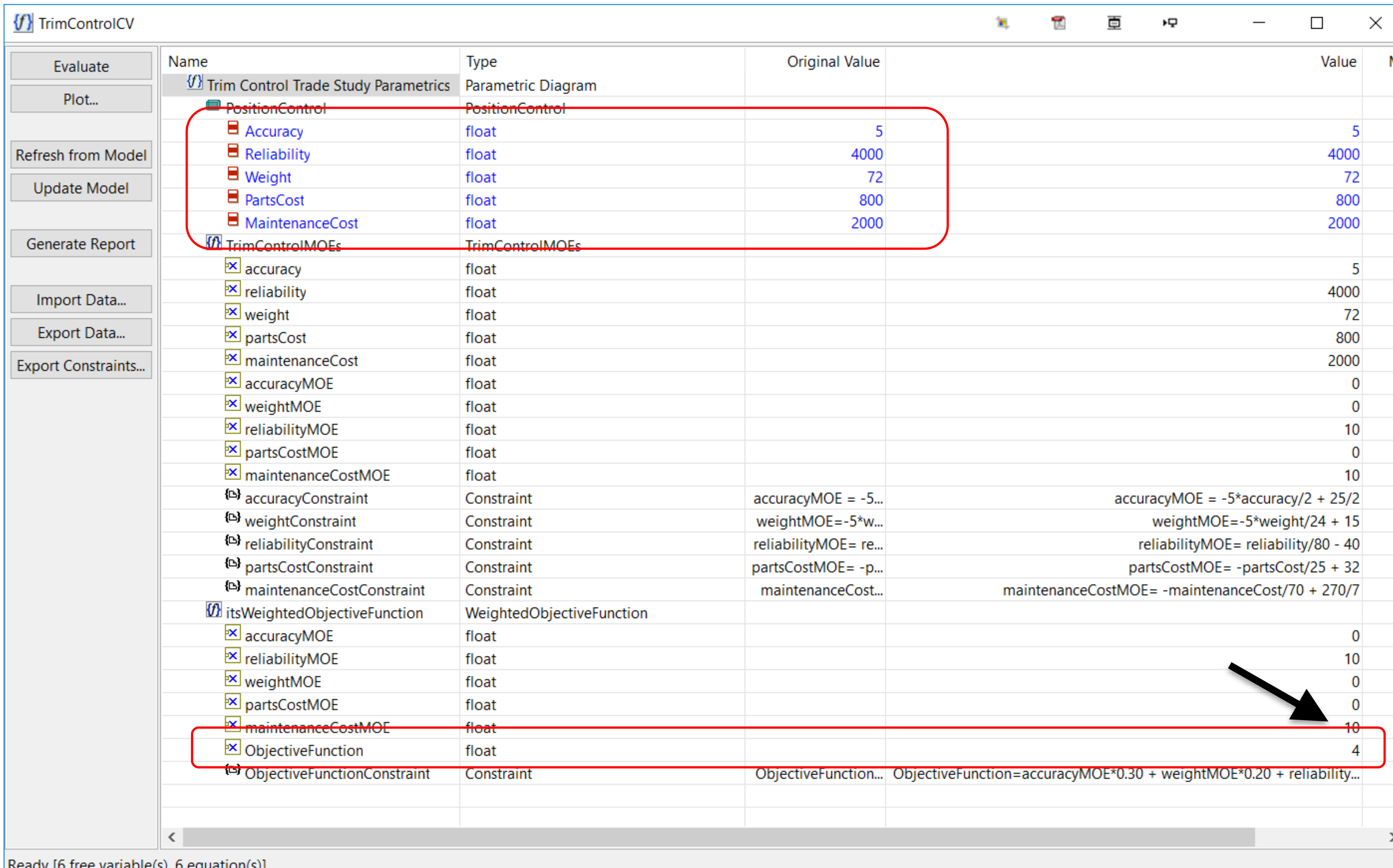
Note: the Objective Function sums up the "goodness" of each criterion weighted by its importance



Architectural Analysis: Evaluate

☑ Option 1

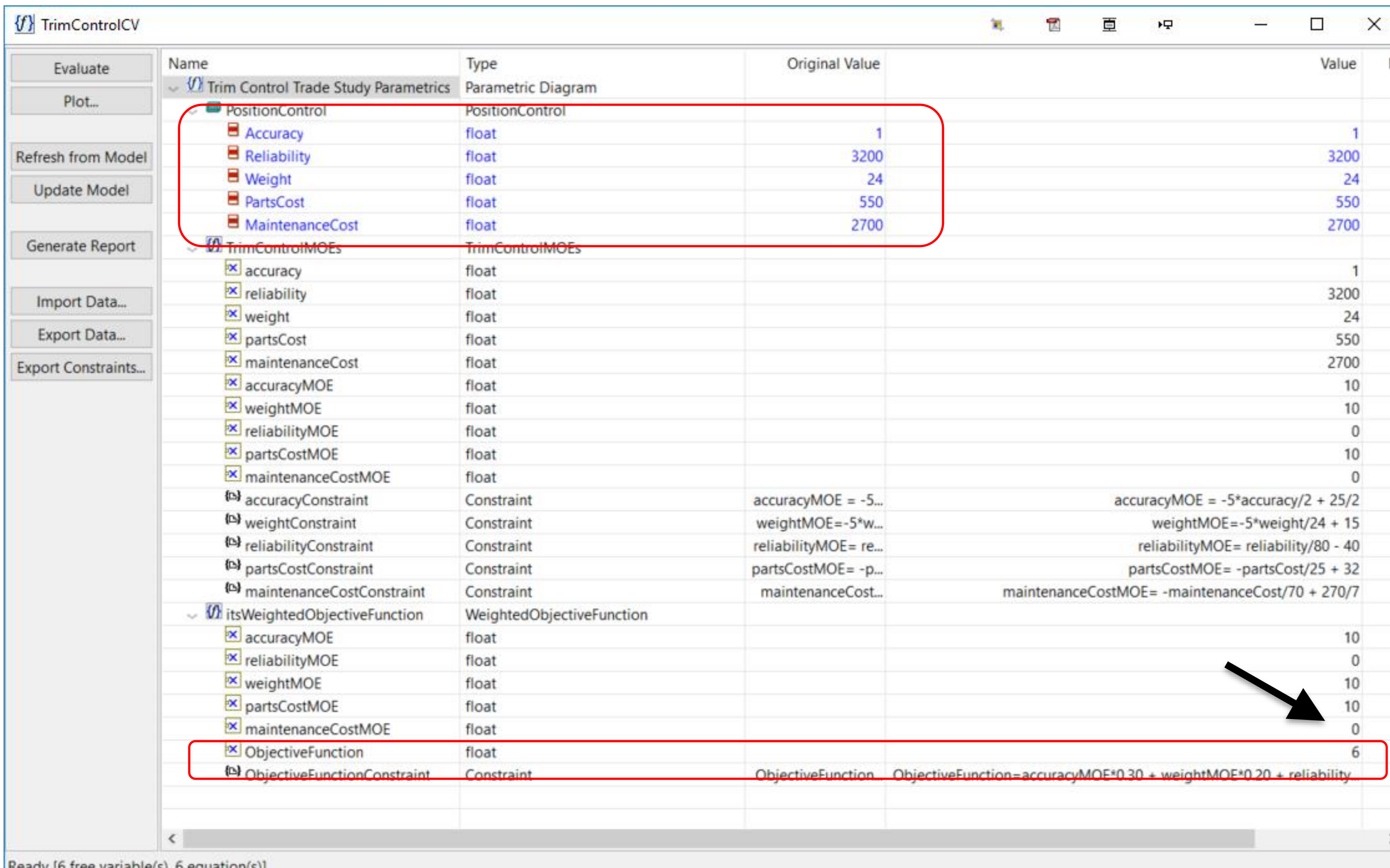
Solution/ moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100



Architectural Analysis: Evaluate

☑ Option 2

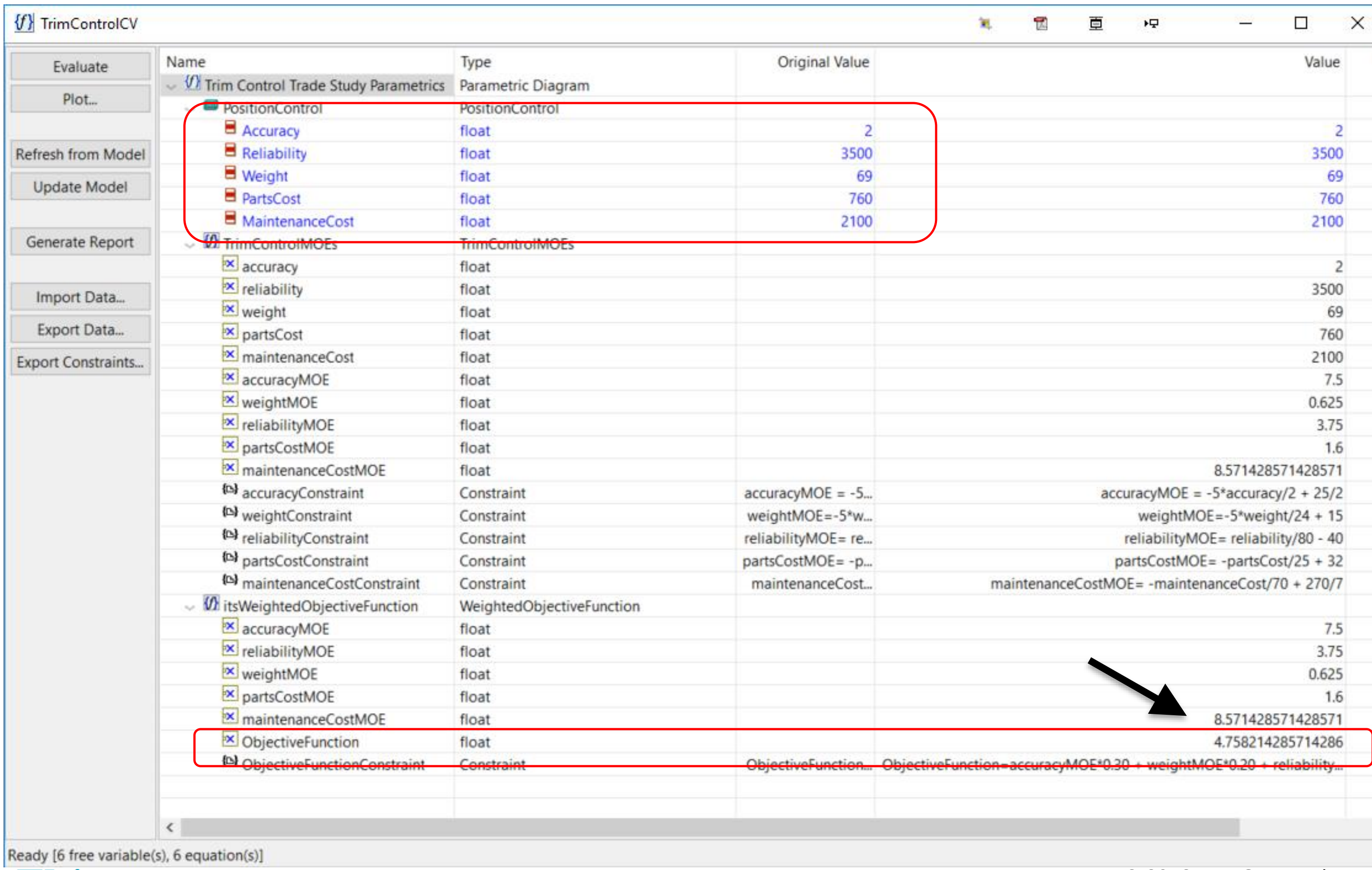
Solution/ moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100



Architectural Analysis: Evaluate

☑ Option 3

Solution/ moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100

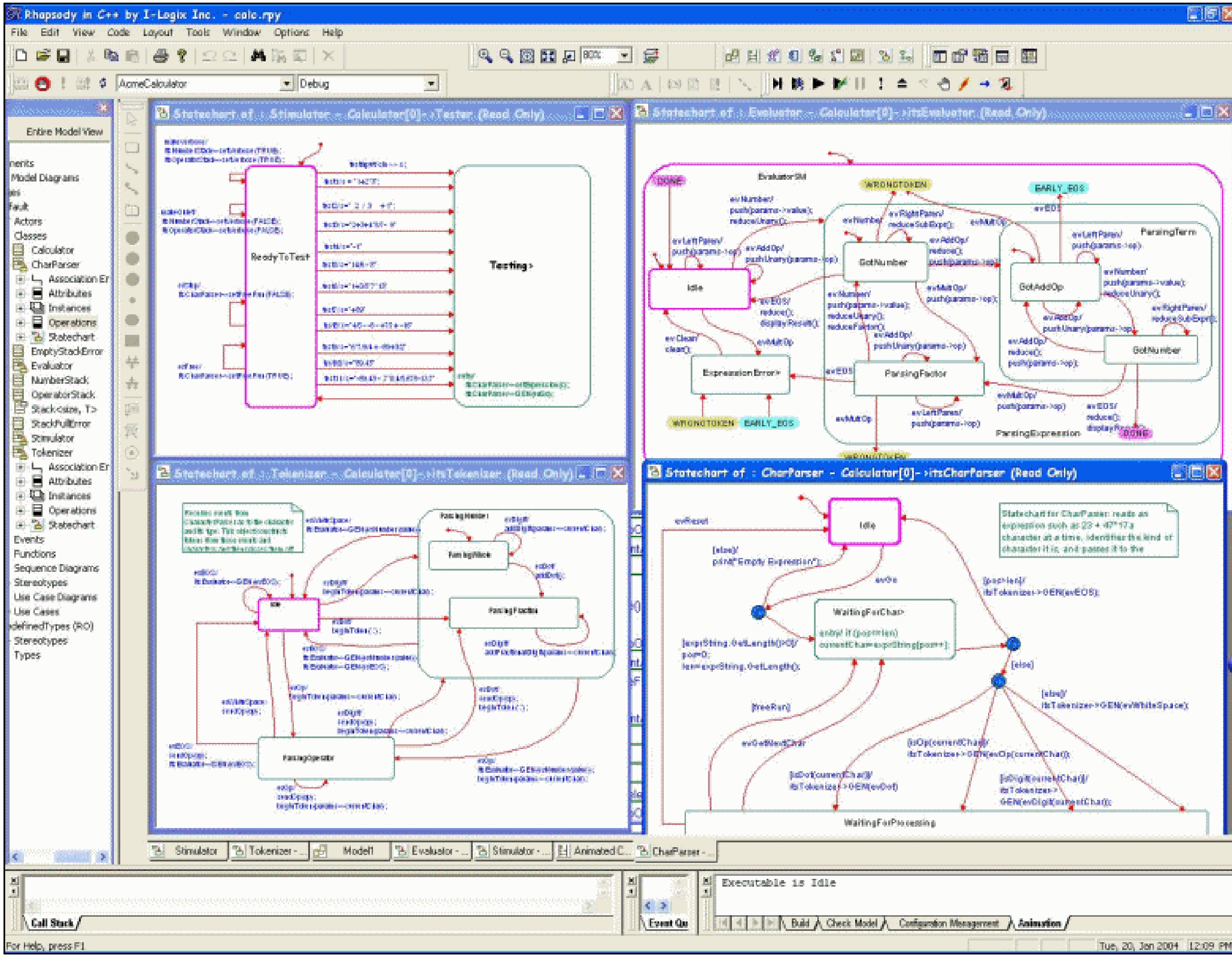


Name	Type	Original Value	Value
Trim Control Trade Study Parametrics	Parametric Diagram		
PositionControl	PositionControl		
Accuracy	float	2	2
Reliability	float	3500	3500
Weight	float	69	69
PartsCost	float	760	760
MaintenanceCost	float	2100	2100
TrimControlMOEs	TrimControlMOEs		
accuracy	float		2
reliability	float		3500
weight	float		69
partsCost	float		760
maintenanceCost	float		2100
accuracyMOE	float		7.5
weightMOE	float		0.625
reliabilityMOE	float		3.75
partsCostMOE	float		1.6
maintenanceCostMOE	float		8.571428571428571
accuracyConstraint	Constraint	accuracyMOE = -5...	accuracyMOE = -5*accuracy/2 + 25/2
weightConstraint	Constraint	weightMOE=-5*w...	weightMOE=-5*weight/24 + 15
reliabilityConstraint	Constraint	reliabilityMOE= re...	reliabilityMOE= reliability/80 - 40
partsCostConstraint	Constraint	partsCostMOE= -p...	partsCostMOE= -partsCost/25 + 32
maintenanceCostConstraint	Constraint	maintenanceCost...	maintenanceCostMOE= -maintenanceCost/70 + 270/7
itsWeightedObjectiveFunction	WeightedObjectiveFunction		
accuracyMOE	float		7.5
reliabilityMOE	float		3.75
weightMOE	float		0.625
partsCostMOE	float		1.6
maintenanceCostMOE	float		8.571428571428571
ObjectiveFunction	float		4.758214285714286
ObjectiveFunctionConstraint	Constraint	ObjectiveFunction...	ObjectiveFunction=accuracyMOE*0.30 + weightMOE*0.20 + reliability...

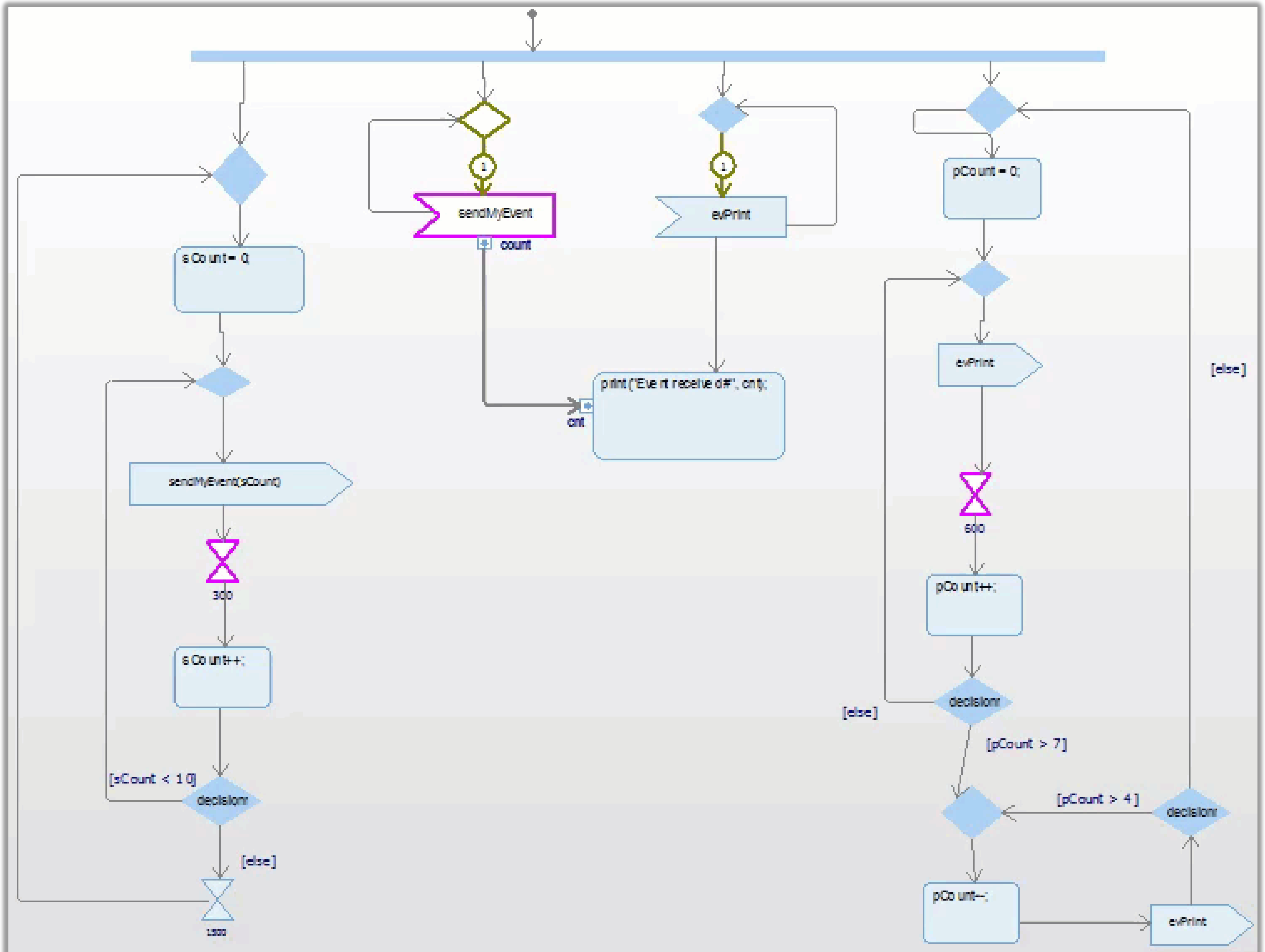
Run-Time Computational Behavior: Executable Models

- Executable Models do computation in a specific direction at run-time. UML/SysML provides behavioral models that can perform computation that takes place a run-time.
- Run-time can be either in a simulation or in an actual developed system
- In addition, Rhapsody can connect to other tools that provide run-time computation, including
  - Simulink
  - Functional Mockup Interface (FMI) tools such as SimulationX or Modelica
- These can be used in a number of different ways, such as
  - Model verification, such as with executable requirements models
  - System simulation with tools providing environmental or physics models
  - Systems with control models done in Simulink
  - Digital Twins combining actual operational data with system simulation (such as for preventative maintenance)

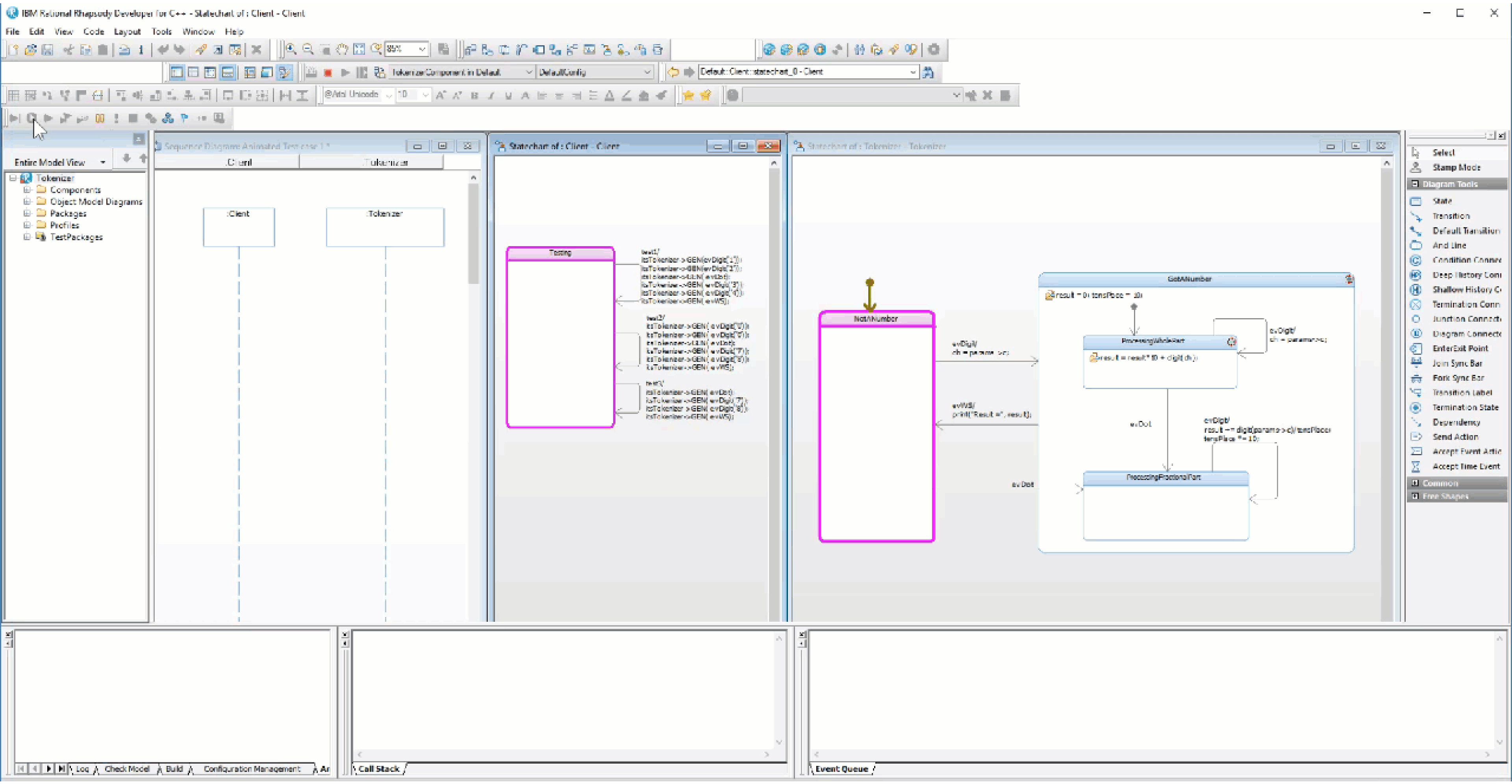
Run-Time Computational Behavior: State Machine



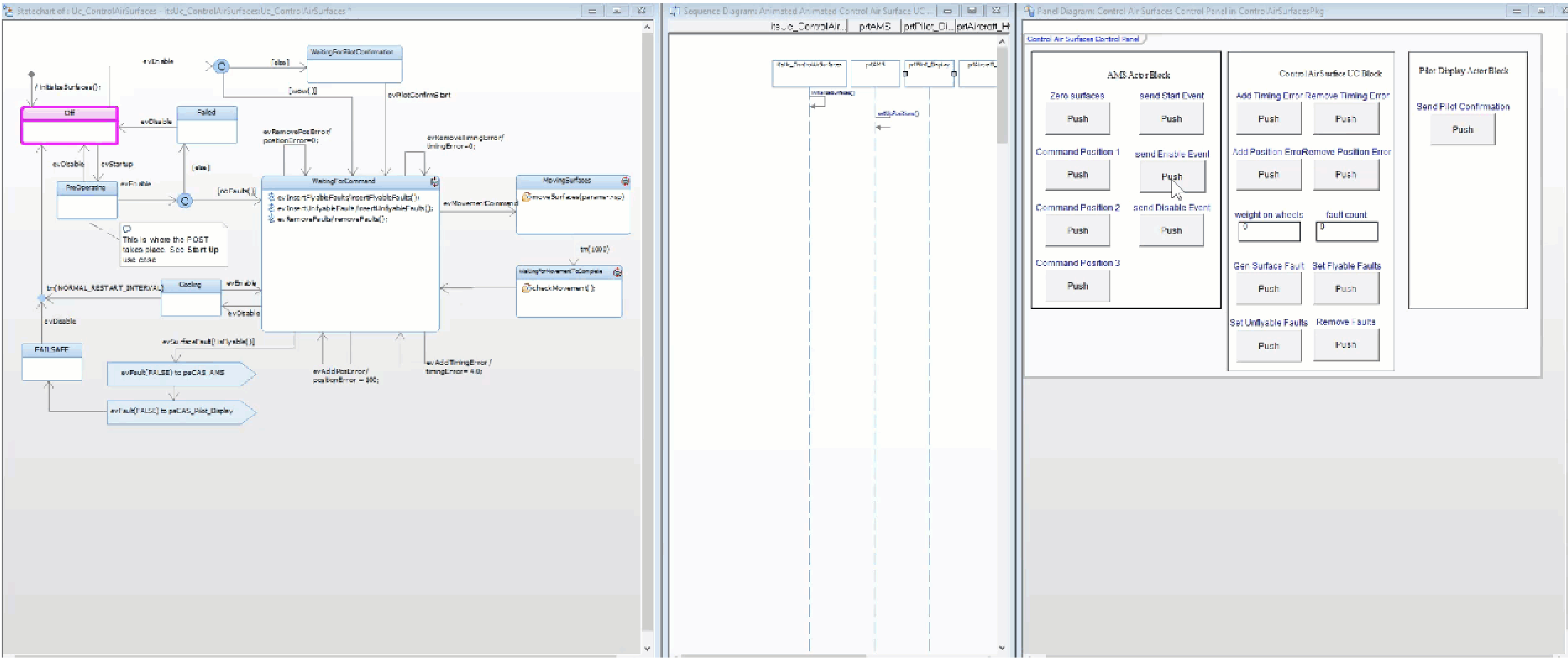
Run-Time Computational Behavior: Activity Diagrams



Run-Time Computational Behavior: Sequence Diagrams

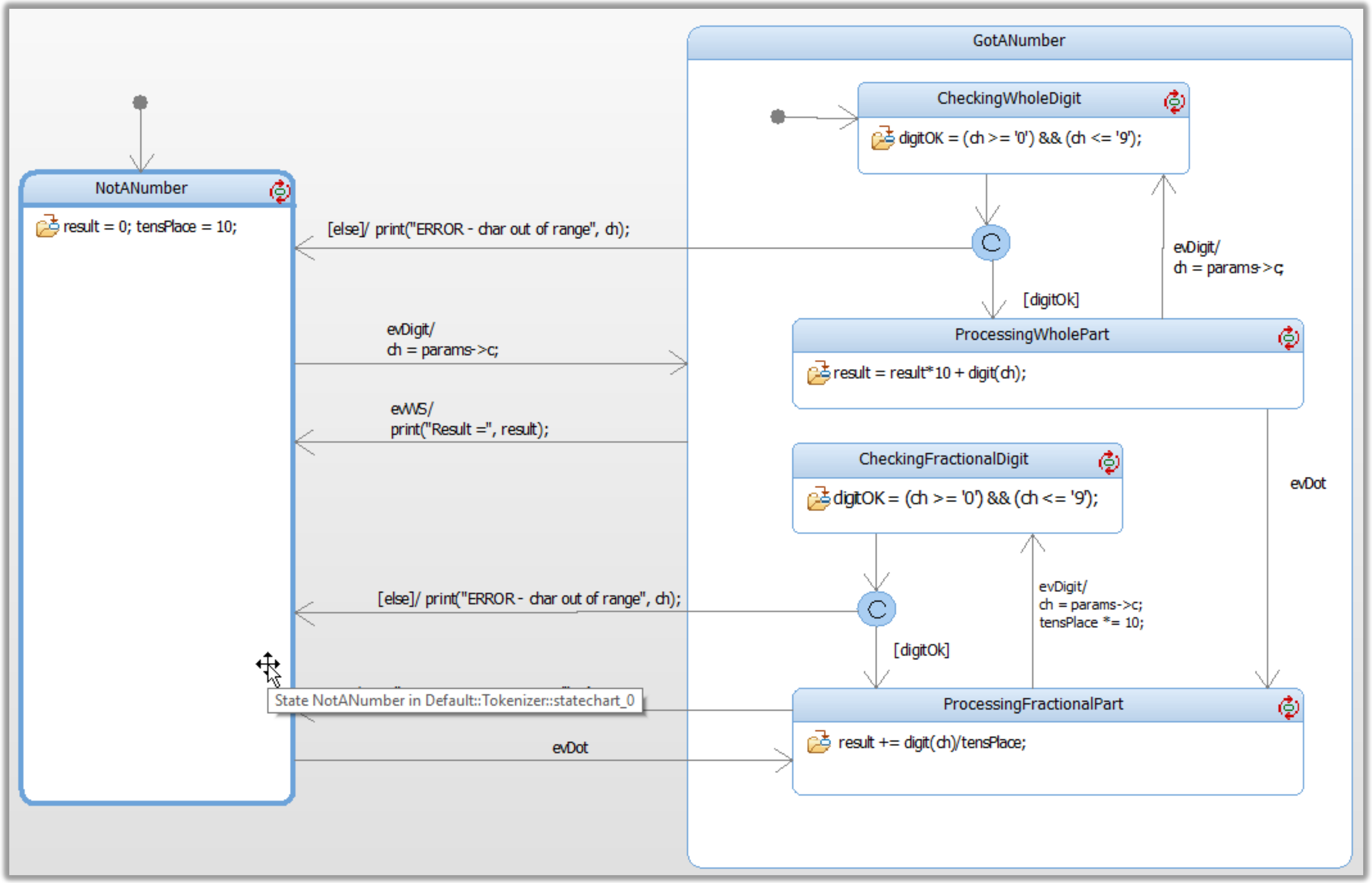


Run-Time Computational Behavior: Panel Diagrams



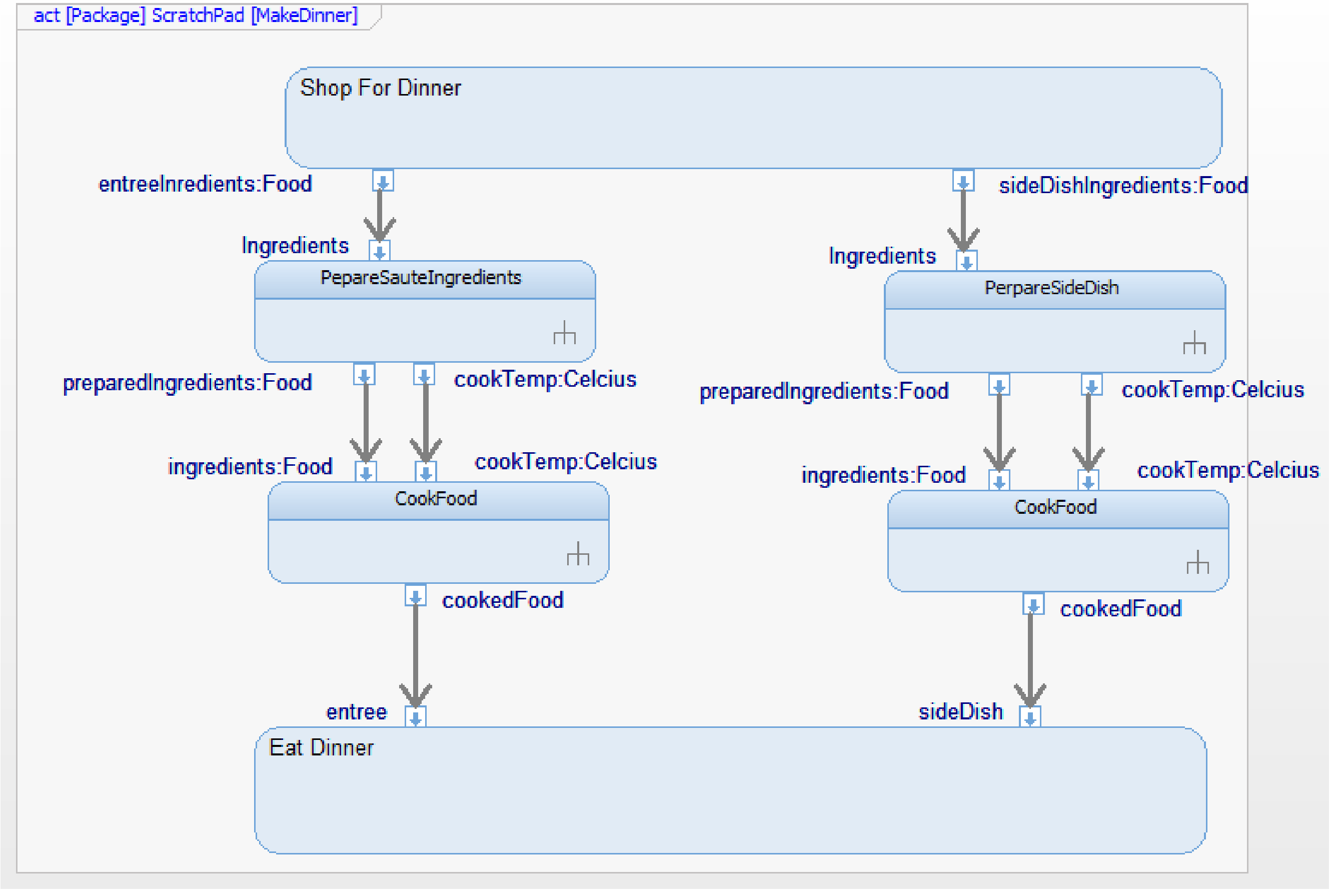
Executable UML/SysML

- SysML behavioral models organize and orchestrate the execution of actions
  - Actions appear as **usages of action specifications** in state diagrams as entry, exit, transition, or internal actions



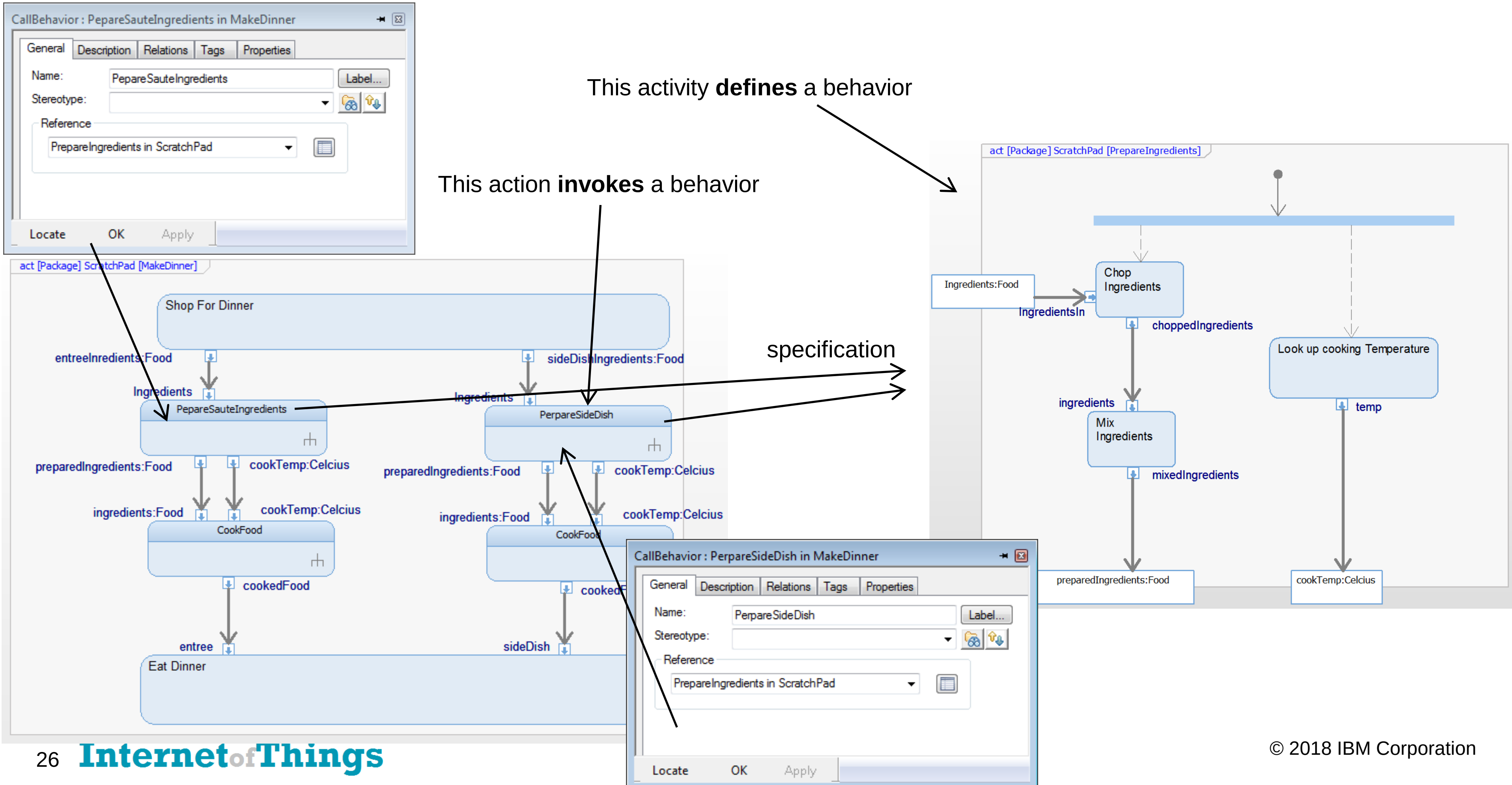
Executable UML/SysML

- SysML behavioral models organize and orchestrate the execution of actions
  - Actions appear in activity diagrams as **usages of action specifications**



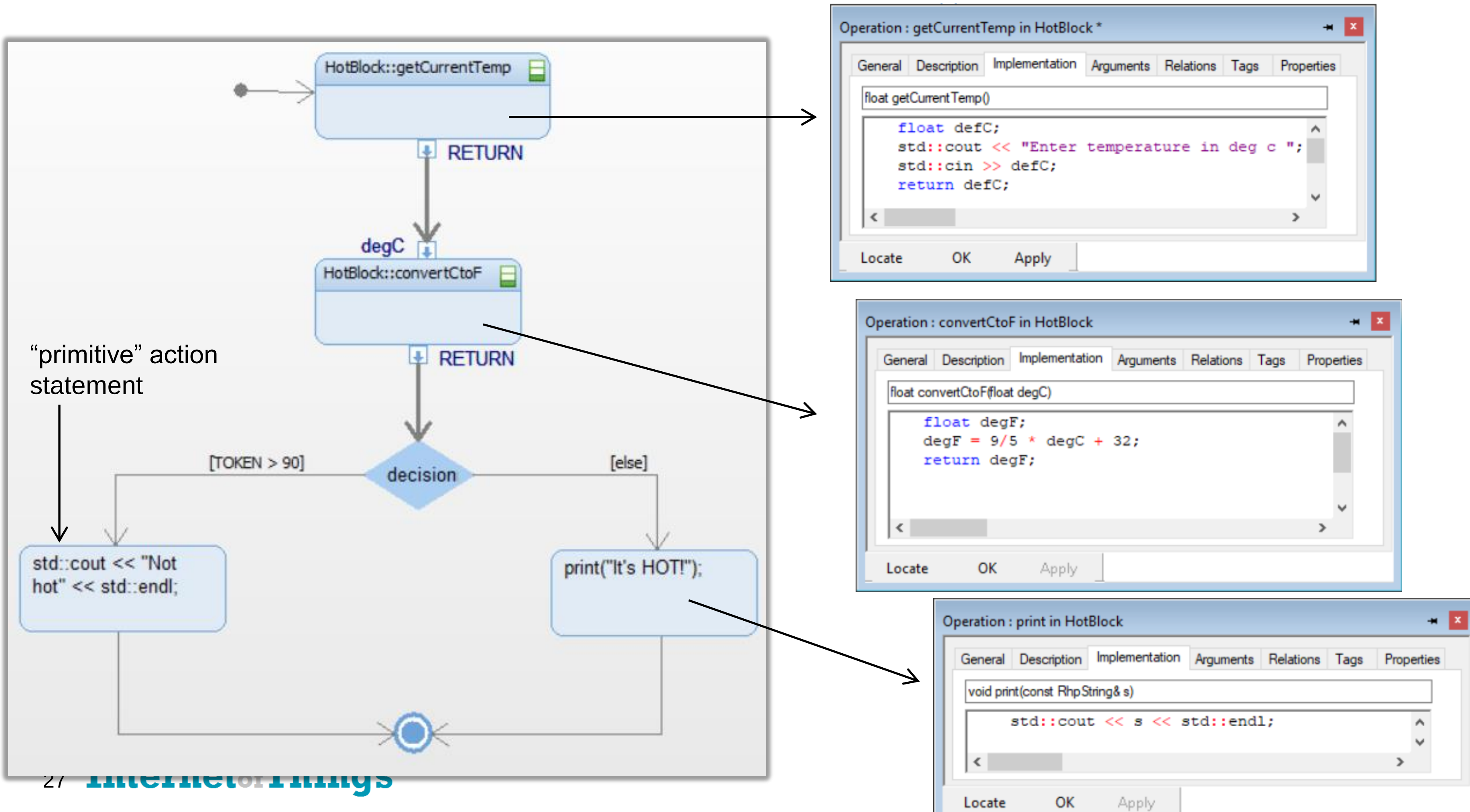
Executable UML/SysML

- SysML behavioral models organize and orchestrate the execution of actions
  - Actions may be specified
    - By activity diagrams

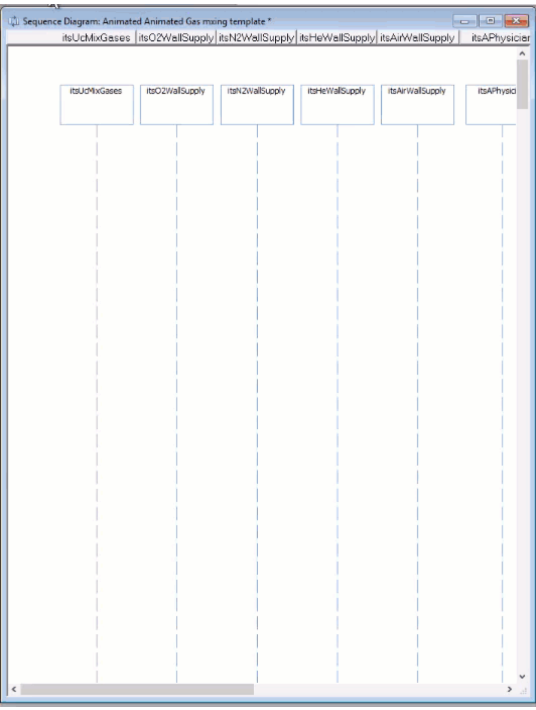


Executable UML/SysML

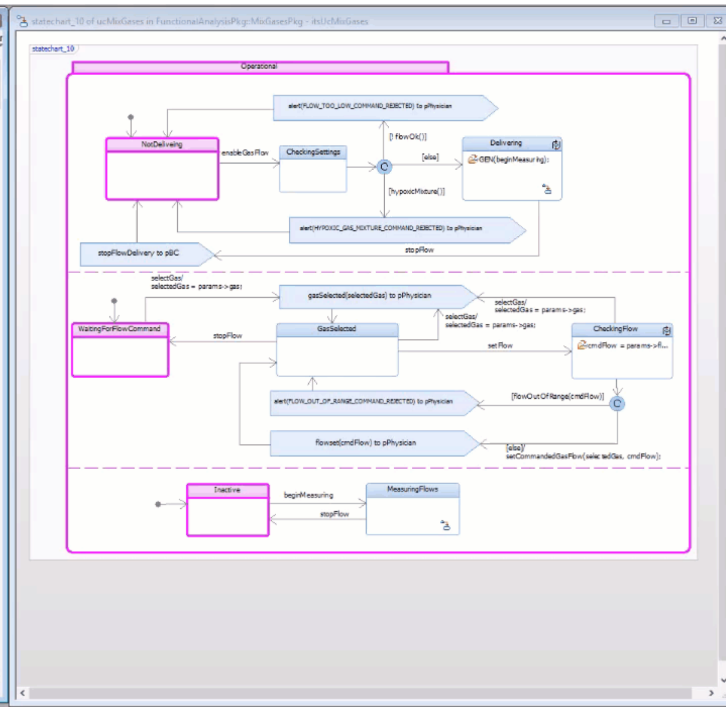
- SysML behavioral models organize and orchestrate the execution of actions
  - Actions may be specified
    - By an “action language” such as C, C++, Ada, or Java



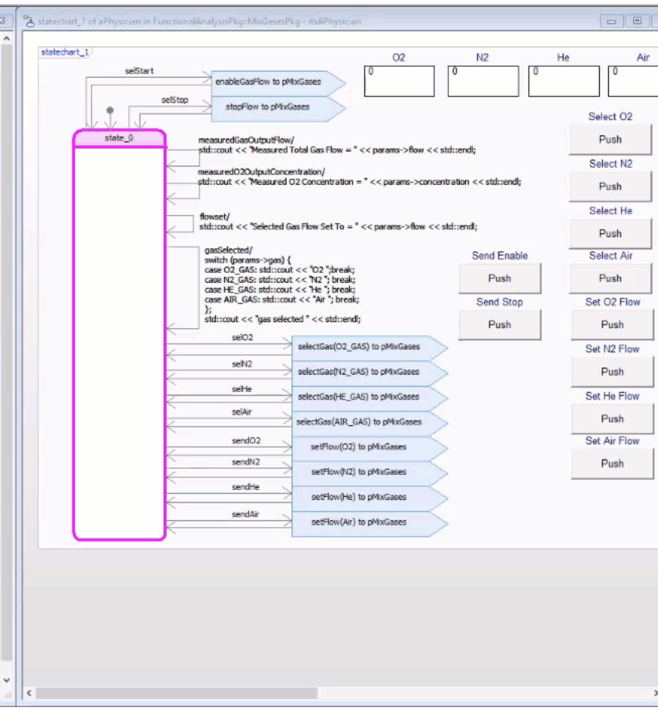
Verification of an Executable Requirements Model



Sequence Diagram (generated)



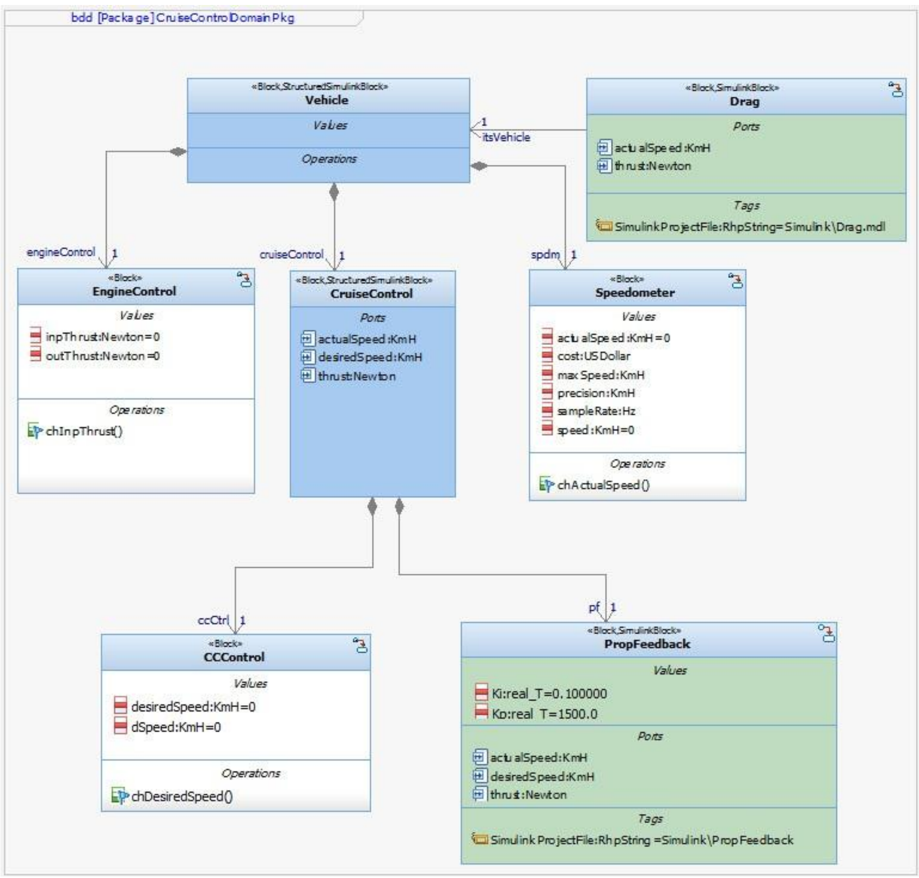
Use Case state machine

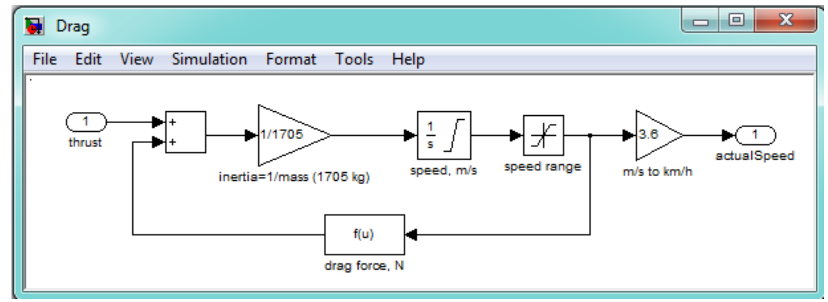


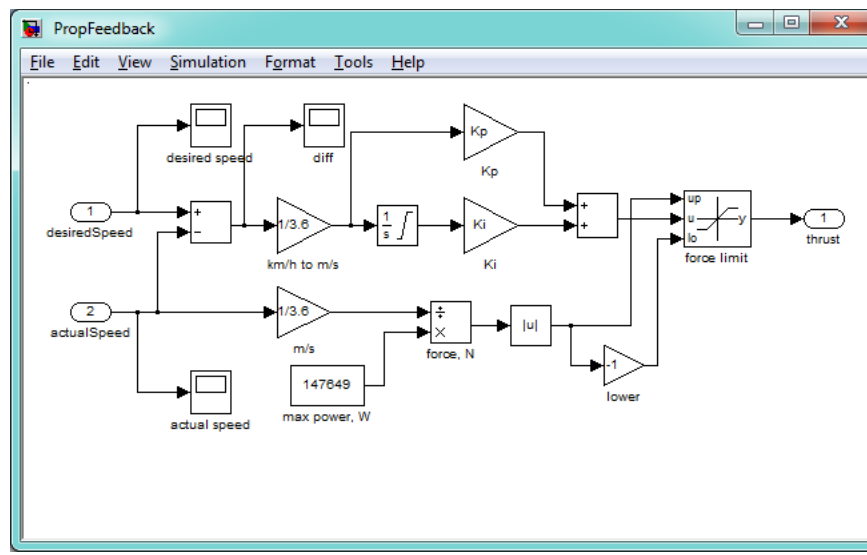
Actor state machine with panel diagram elements

Cosimulation with SysML: «SimulinkBlock»

- The stereotype «SimulinkBlock» means the block’s behavior is specified in a Simulink model
- Every input/output port in the Simulink model is represented as a SysML atomic flow port
- Type matching rules need to be applied

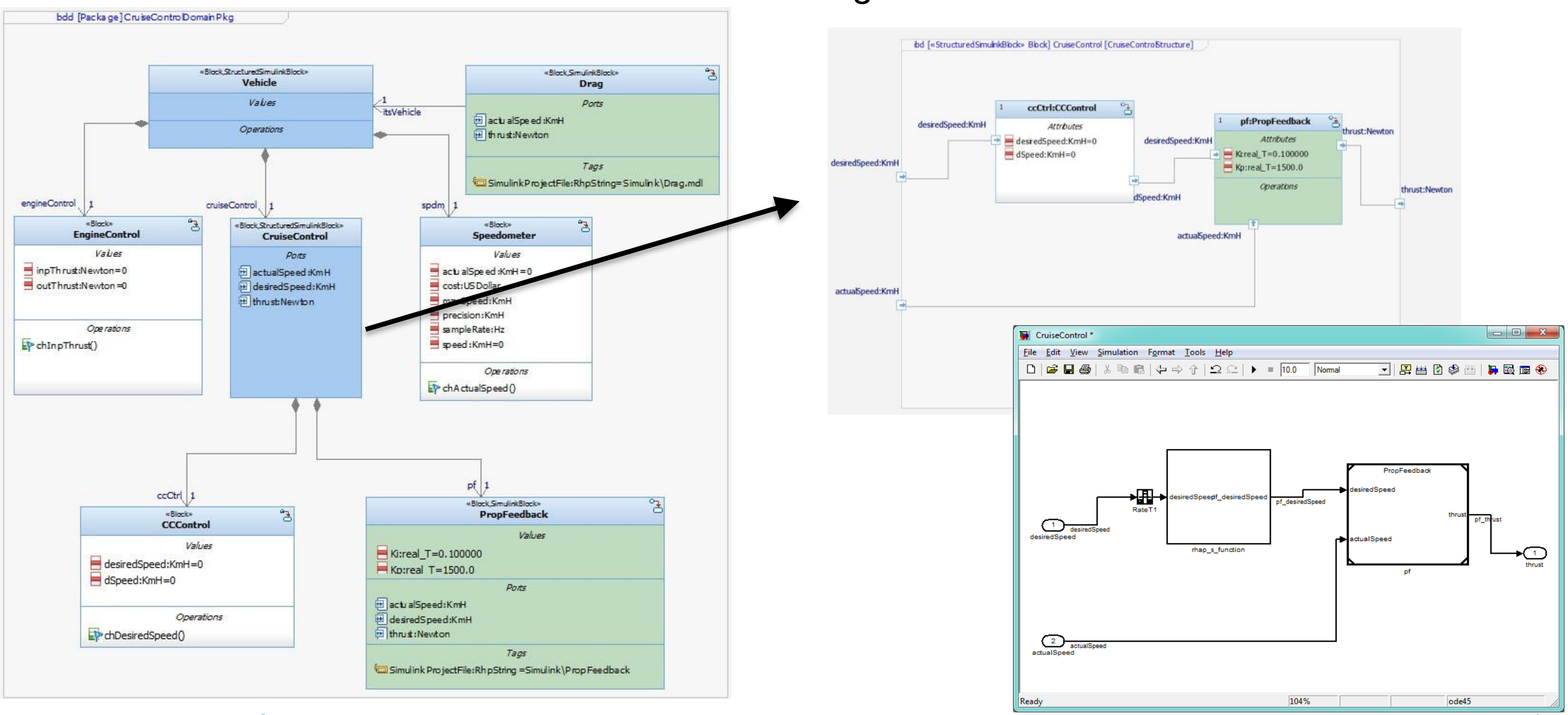






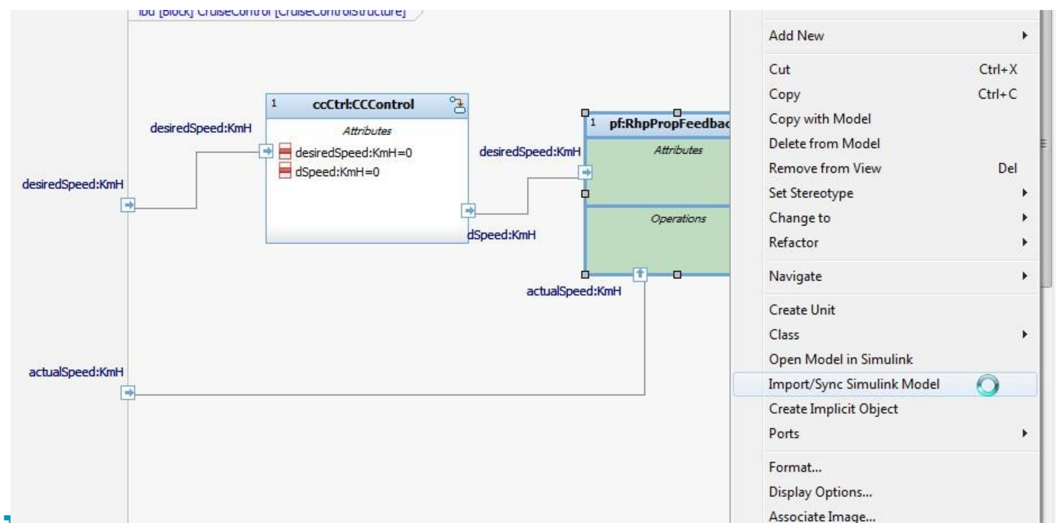
Cosimulation with SysML: «StructuredSimulinkBlock»

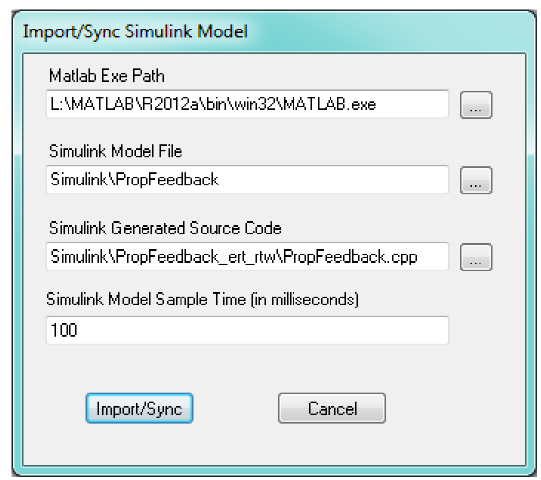
- The stereotype «StructuredSimulinkBlock» means the block has parts typed by Simulink blocks
  – A block that owns a part typed by a «StructuredSimulinkBlock» is also a «StructuredSimulinkBlock»
- A «StructuredSimulinkBlock» can be exported to Simulink for simulation
  – All non Simulink blocks are transformed to a single S-Function in Simulink



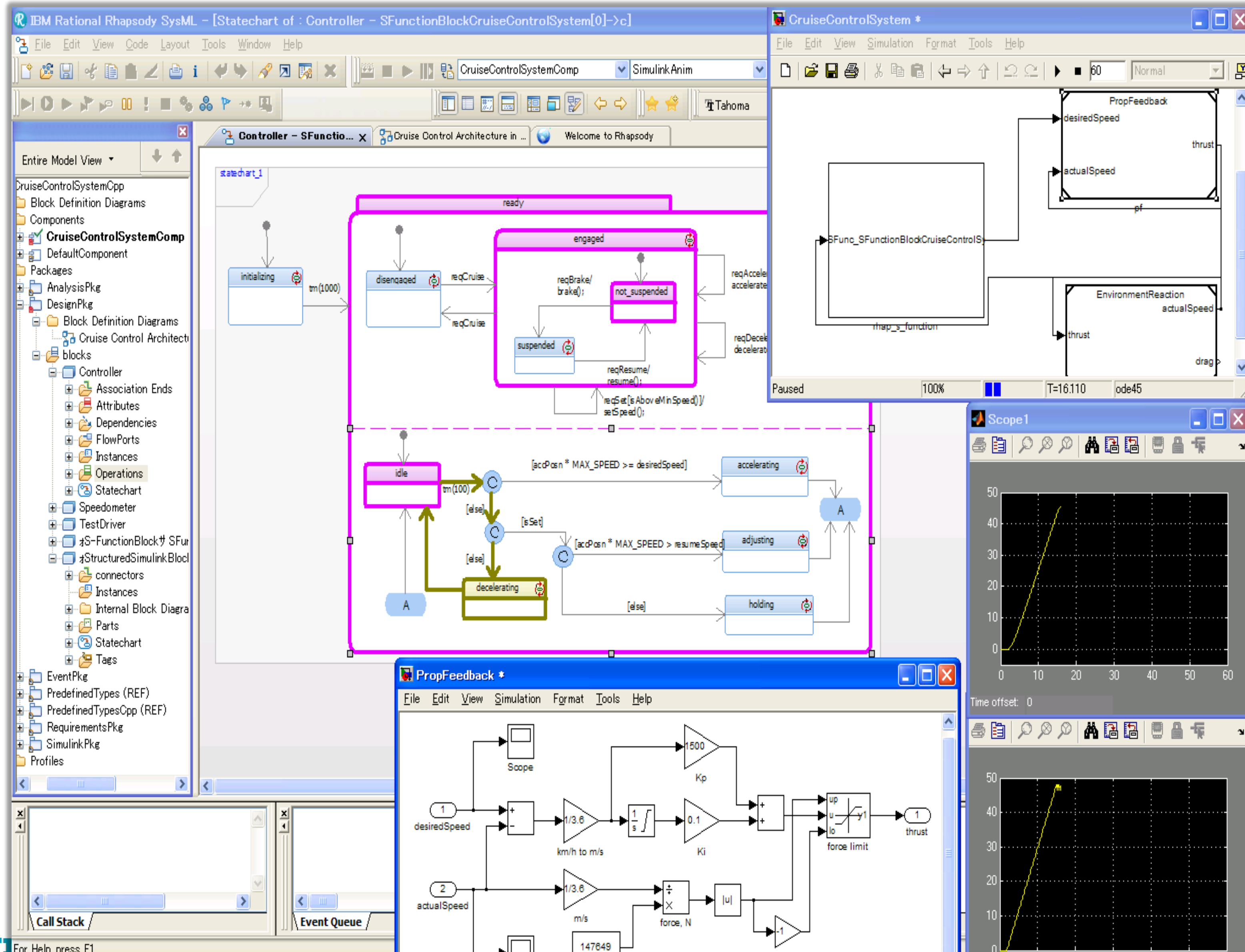
Exchanging behavior via generated code

- Our approach uses generated C/C++ code to generate behavior of blocks brought to the simulator
- «SimulinkBlock» may reference C/C++ code generated by MATLAB Embedded Coder
  - This code is compiled with the rest of the code into an executable used by Rhapsody simulation
- «StructuredSimulinkBlock» is transformed to a Simulink model with an auto-generated S-Function Block that encapsulates the behavior of the native SysML blocks
- Modelica has adopted the Functional Mockup Interface (FMI) standard (see https://www.fmi-standard.org/) to exchange behavior using generated C code
  - Unlike S-Function, FMI is non-proprietary





Flow ports are used to connect to Simulink for co-simulation



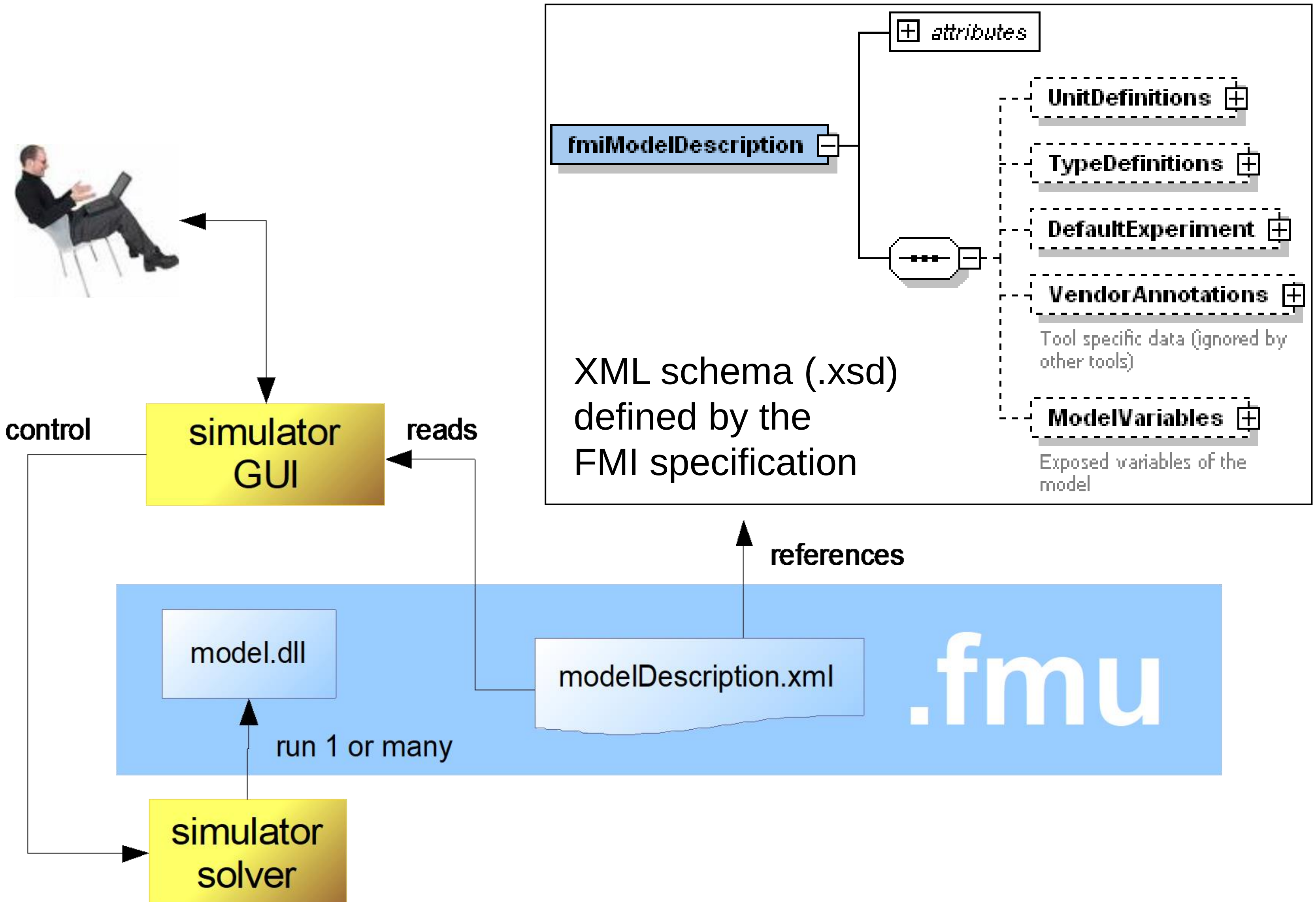
FMI Standard

The FMI development was part of the ITEA2 MODELISAR project (2008 - 2011; 29 partners, Budget: 30 Mill. €). From 2012 FMI is developed as Modelica Association project

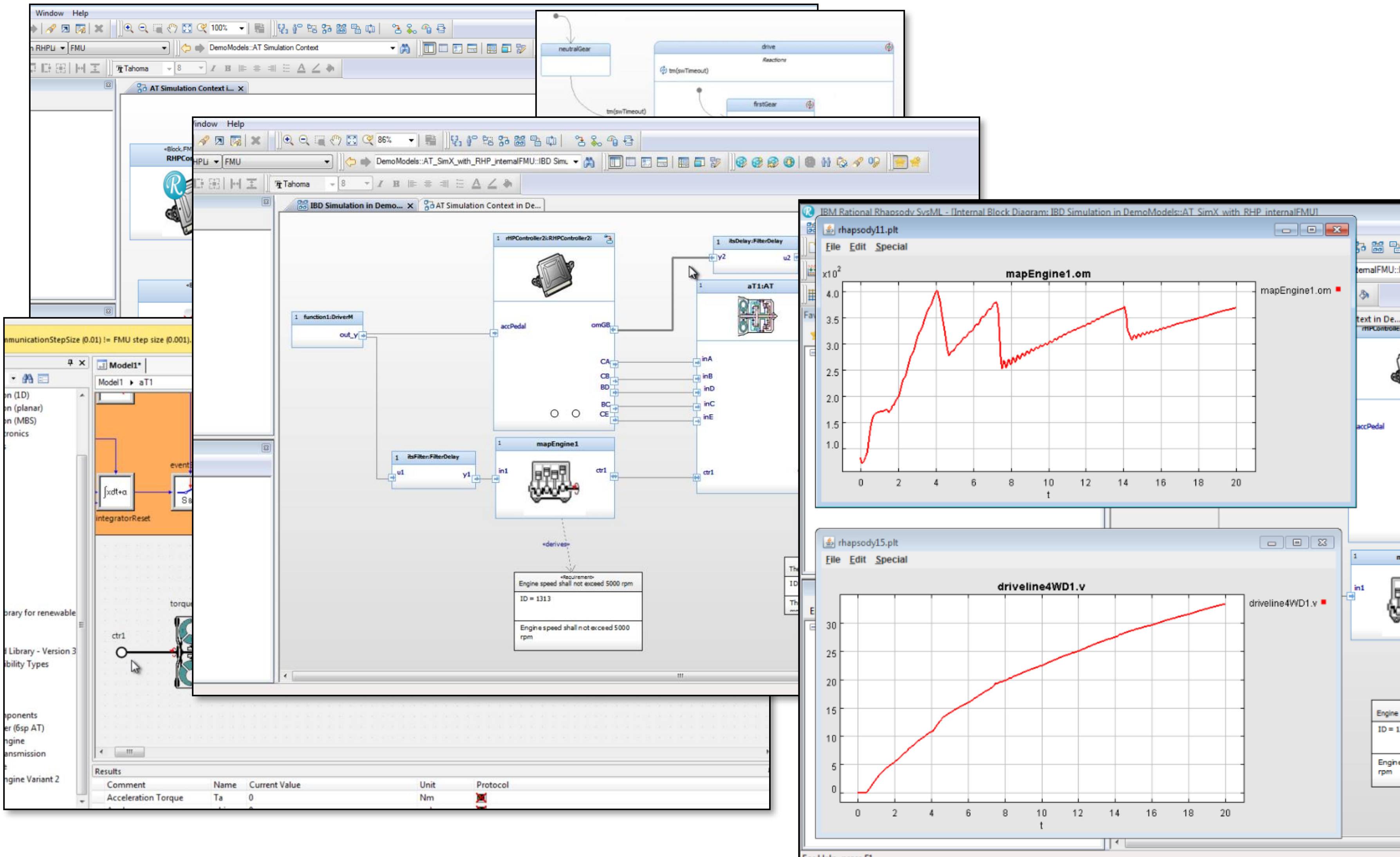
- FMI development initiated, organized and headed by Daimler AG
- Improved Software/Model/Hardware-in-the-Loop Simulation, of physical models from different vendors.
- Open Standard
- FMI Standard Releases
  - FMI 1.0 in 2010
  - FMI 2.0 in 2014
- Over 35 FMI compliant tools (Modelica tools, Simulink add-ons, Rhapsody, etc)
  - https://www.fmi-standard.org/tools

functional mockup interface for model exchange and tool coupling

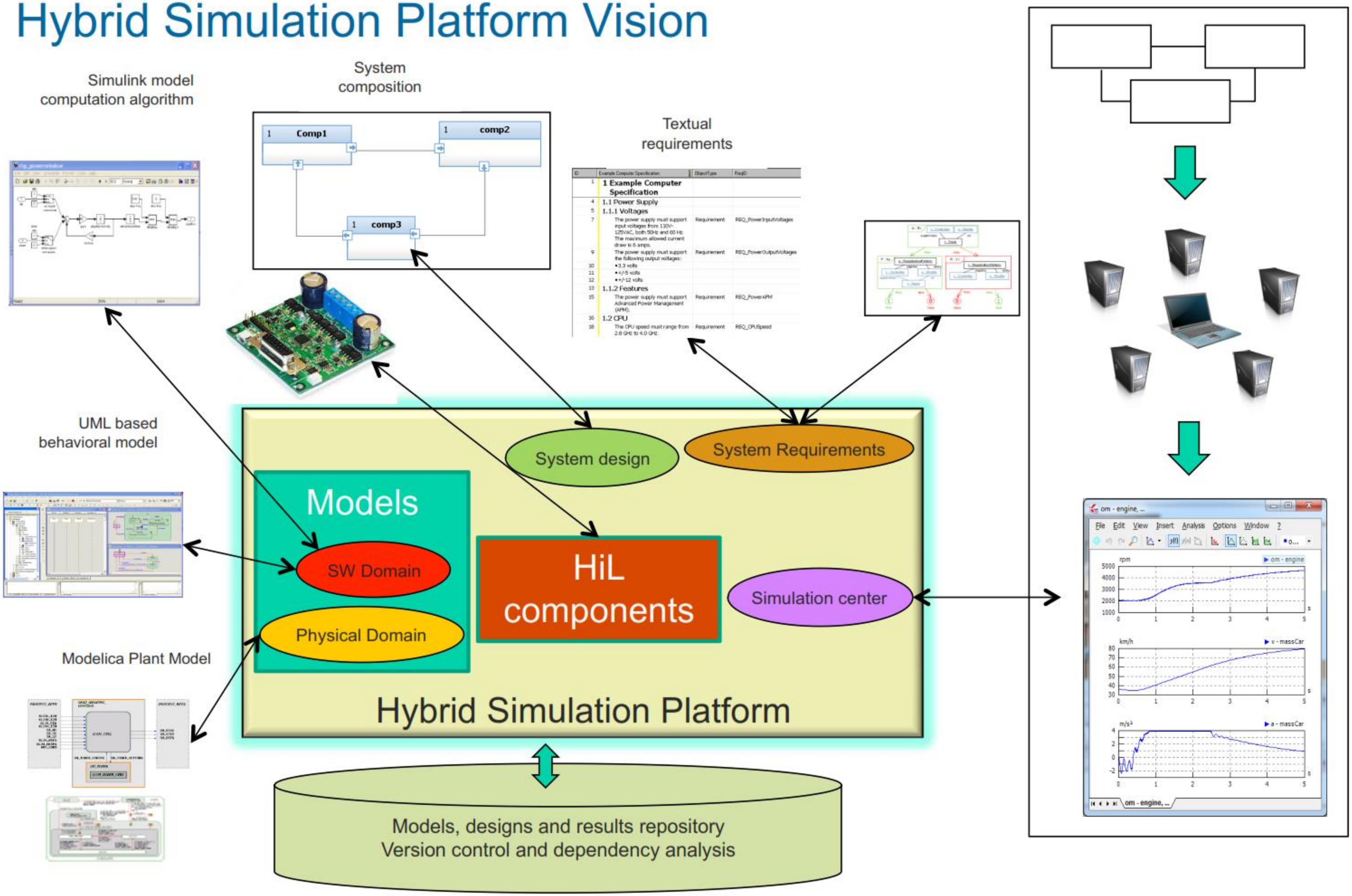
Function Mockup Unit (FMU)



FMU SimulationX / IBM Rhapsody Integration



Hybrid Simulation Platform Vision



Download Papers, Presentations, Models, & Profiles for Free