Use Cases For Model Execution

Žilvinas Strolia, 2016 May

No Magic

The Truth is in the Models™

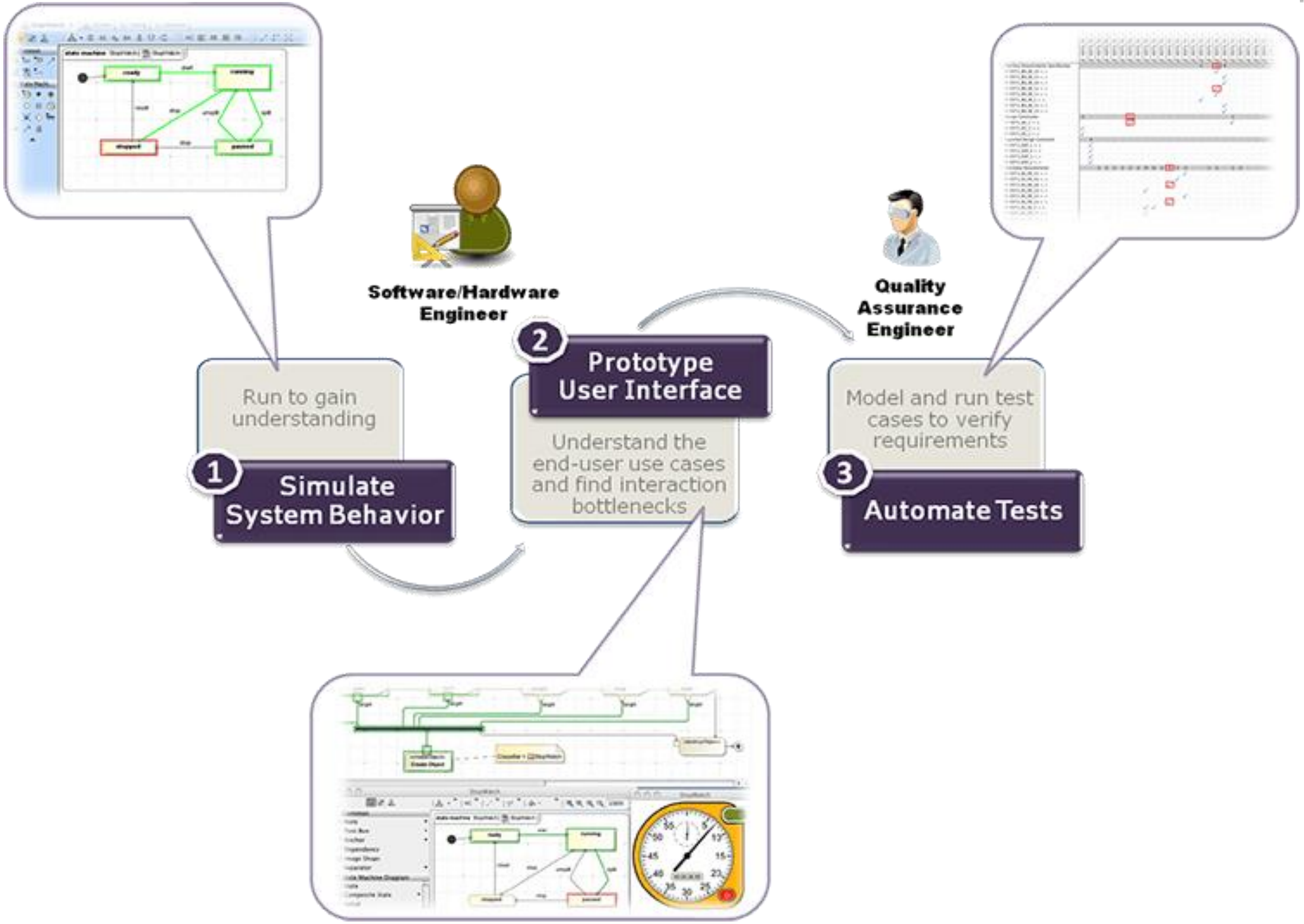
Outline

- Introduction
- Automation and Debugging
- Behavior Simulation
- User Interface Mockups
- Engineering analysis
  - Automated Requirements Verification
  - Trade studies / trade-off analysis
  - Total mass/power/cost rollups
  - Model-based testing
- Integration of Analytics Models

Introduction

Simulation

- The purpose of a simulation is to gain system understanding without manipulating the real system, either because it is not yet defined or available, or because it cannot be exercised directly due to cost, time, resources or risk constraints.
- Simulation is typically performed on a model of the system.



Software/Hardware Engineer

Quality Assurance Engineer

1 Simulate System Behavior

Run to gain understanding

2 Prototype User Interface

Understand the end-user use cases and find interaction bottlenecks

3 Automate Tests

Model and run test cases to verify requirements

Cameo Simulation Toolkit (CST)

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

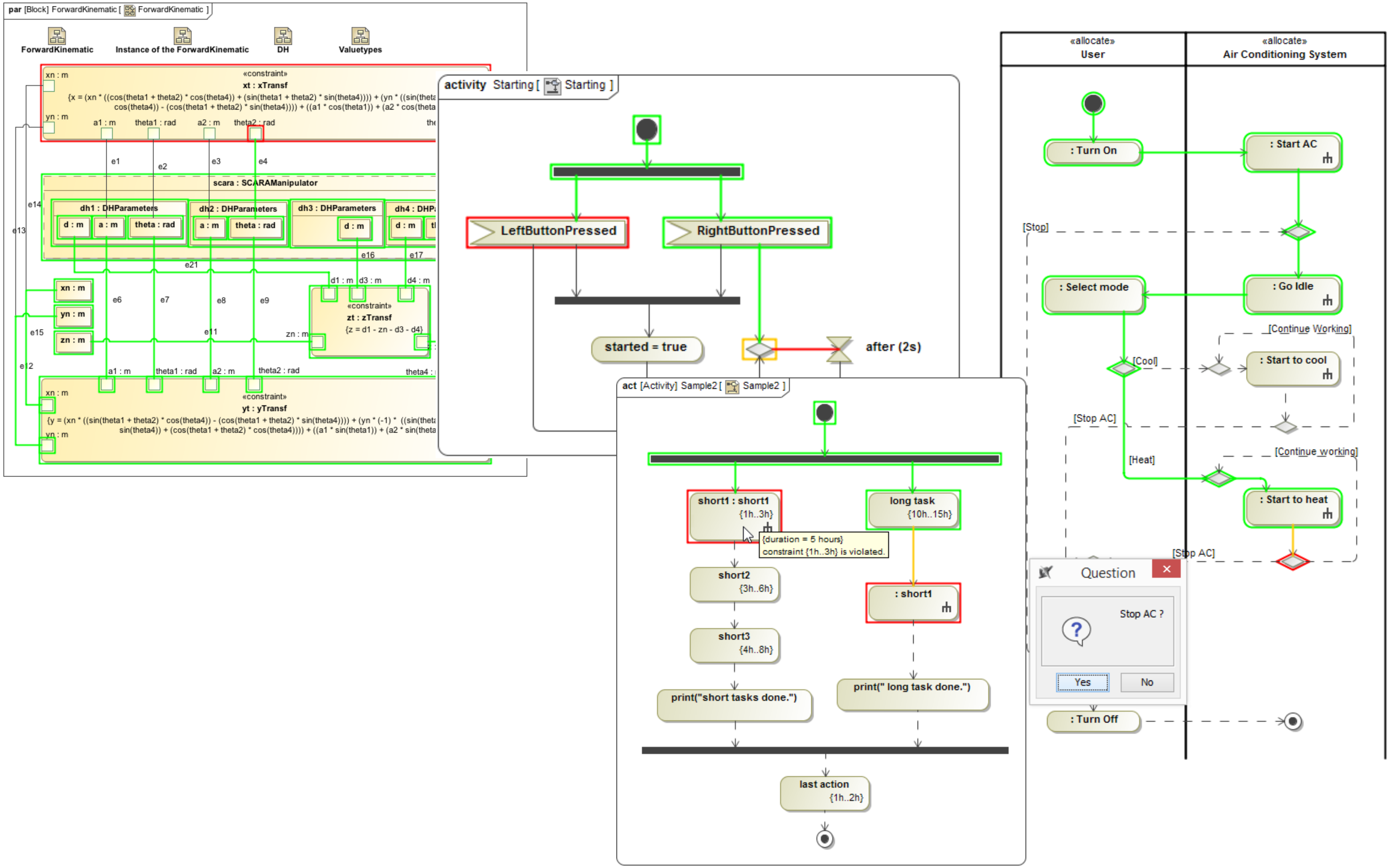
- The standard based model execution of:
  - Activities (OMG fUML standard)
  - Composite structures (OMG PSCS)
  - State Machines (W3C SCXML standard)
  - Actions/scripts (JSR223 standard)
  - Parametrics (OMG SysML standard)
  - Sequence diagrams (OMG UML Testing Profile)

Animation and debugging

Enhance user understanding

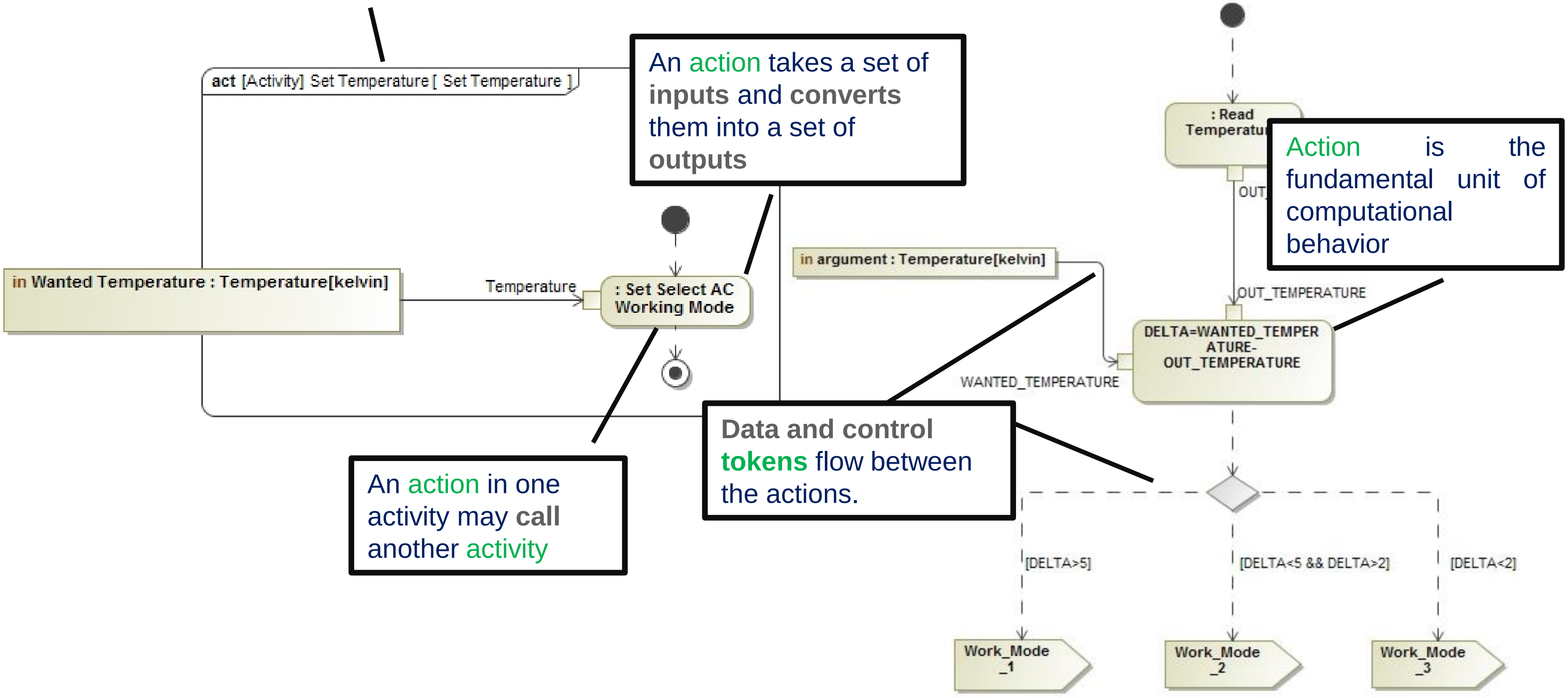
- Animating various diagrams during the execution of a dynamic system model representing the system behavior can significantly enhance user understanding.
- A simple simulation can either rely on execution of pre-scripted scenarios, or it can react to specific user interaction (e.g., “toggle this input and see what happens”).

Animation


par [Block] ForwardKinematic [ForwardKinematic]
ForwardKinematic
Instance of the ForwardKinematic
DH
Valuetypes
xn : m
yn : m
«constraint»
xt : xTransf
{x = (xn * ((cos(theta1 + theta2) * cos(theta4)) + (sin(theta1 + theta2) * sin(theta4)))) + (yn * ((sin(theta
cos(theta4)) - (cos(theta1 + theta2) * sin(theta4)))) + ((a1 * cos(theta1)) + (a2 * cos(theta
a1 : m
theta1 : rad
a2 : m
theta2 : rad
e1
e2
e3
e4
scara : SCARAManipulator
dh1 : DHParameters
dh2 : DHParameters
dh3 : DHParameters
dh4 : DHP
d : m
a : m
theta : rad
e13
e14
e15
e16
e17
e21
e6
e7
e8
e9
e11
e12
d1 : m
d3 : m
d4 : m
«constraint»
zt : zTransf
{z = d1 - zn - d3 - d4}
zn : m
theta4 :
«constraint»
yt : yTransf
{y = (xn * ((sin(theta1 + theta2) * cos(theta4)) - (cos(theta1 + theta2) * sin(theta4)))) + (yn * (-1) * ((sin(theta
sin(theta4)) + (cos(theta1 + theta2) * cos(theta4)))) + ((a1 * sin(theta1)) + (a2 * sin(theta
activity Starting [Starting]
LeftButtonPressed
RightButtonPressed
started = true
after (2s)
act [Activity] Sample2 [Sample2]
short1 : short1
{1h..3h}
{duration = 5 hours}
constraint {1h..3h} is violated.
long task
{10h..15h}
short2
{3h..6h}
short3
{4h..8h}
: short1
print("short tasks done.")
print(" long task done.")
last action
{1h..2h}
«allocate»
User
«allocate»
Air Conditioning System
: Turn On
: Start AC
[Stop]
: Select mode
: Go Idle
[Continue Working]
[Cool]
: Start to cool
[Stop AC]
[Heat]
[Continue working]
: Start to heat
[Stop AC]
Question
Stop AC ?
Yes
No
: Turn Off

Behavior Simulation

Executing Activity Diagrams


act [Activity] Set Temperature [Set Temperature]
in Wanted Temperature : Temperature[kelvin]
Temperature
: Set Select AC Working Mode
in argument : Temperature[kelvin]
: Read Temperature
OUT_
OUT_TEMPERATURE
DELTA=WANTED_TEMPERATURE-OUT_TEMPERATURE
WANTED_TEMPERATURE
[DELTA>5]
[DELTA<5 && DELTA>2]
[DELTA<2]
Work_Mode_1
Work_Mode_2
Work_Mode_3

An activity specifies **behavior** as the execution of a set of actions

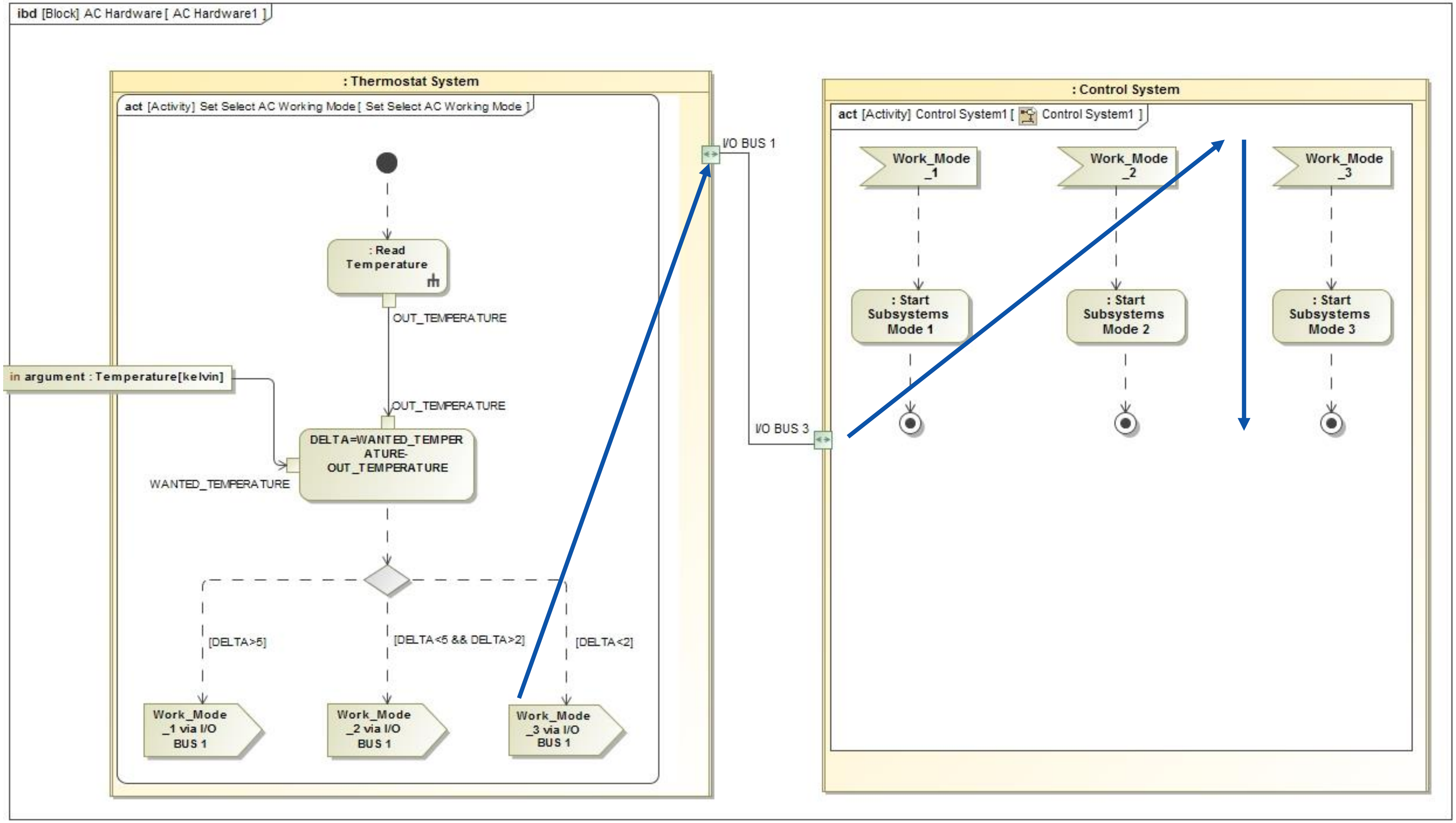
An action takes a set of **inputs and converts** them into a set of **outputs**

Action is the fundamental unit of computational behavior

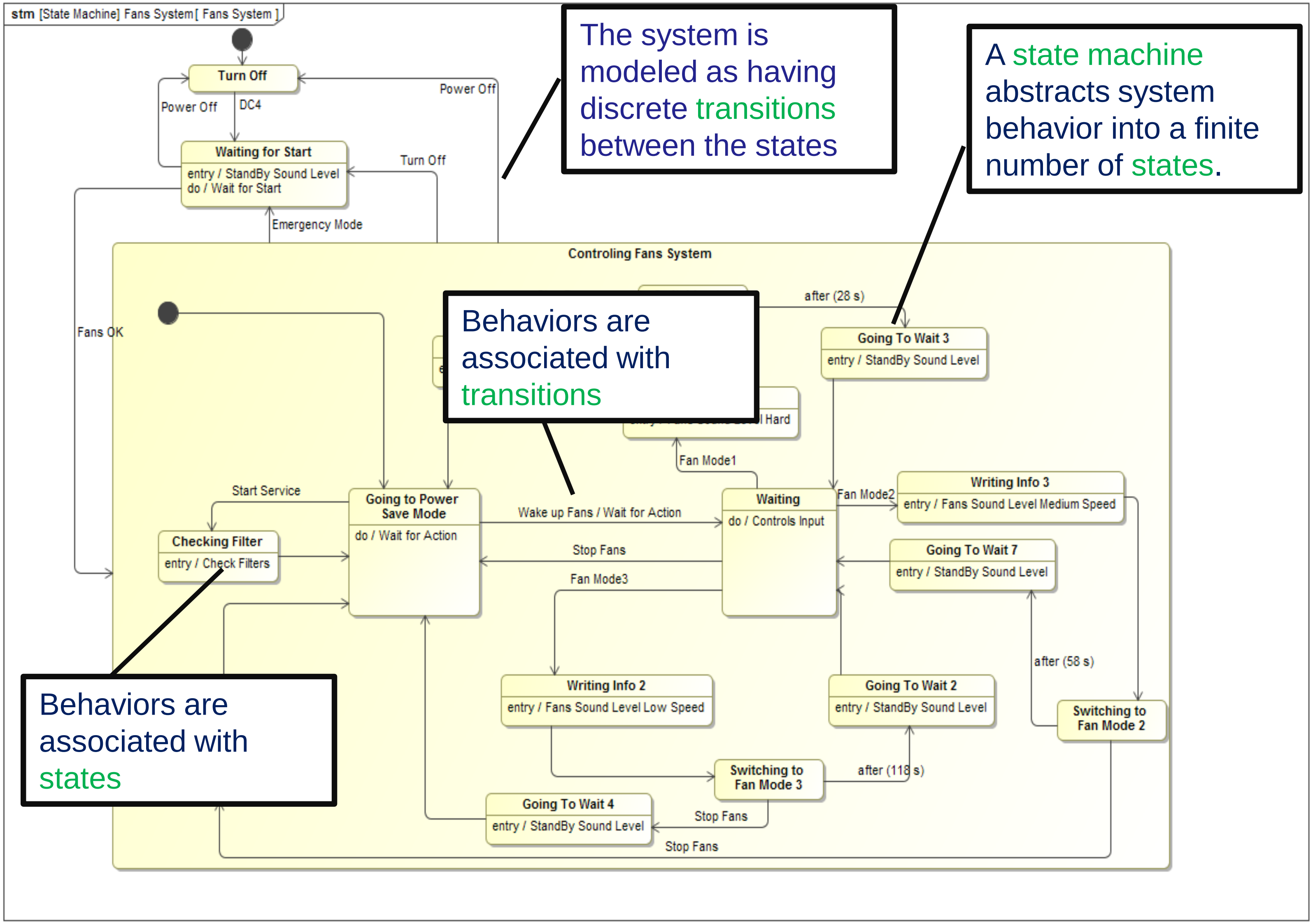
An action in one activity may **call** another activity

Data and control tokens flow between the actions.

Sending Signals via Ports


ibd [Block] AC Hardware [AC Hardware1]
: Thermostat System
act [Activity] Set Select AC Working Mode [Set Select AC Working Mode]
: Read Temperature
OUT_TEMPERATURE
OUT_TEMPERATURE
in argument : Temperature[kelvin]
DELTA=WANTED_TEMPERATURE-OUT_TEMPERATURE
WANTED_TEMPERATURE
[DELTA>5]
[DELTA<5 && DELTA>2]
[DELTA<2]
Work_Mode_1 via I/O BUS 1
Work_Mode_2 via I/O BUS 1
Work_Mode_3 via I/O BUS 1
I/O BUS 1
I/O BUS 3
: Control System
act [Activity] Control System1 [Control System1]
Work_Mode_1
Work_Mode_2
Work_Mode_3
: Start Subsystems Mode 1
: Start Subsystems Mode 2
: Start Subsystems Mode 3

Executing State Machines



The system is modeled as having discrete transitions between the states

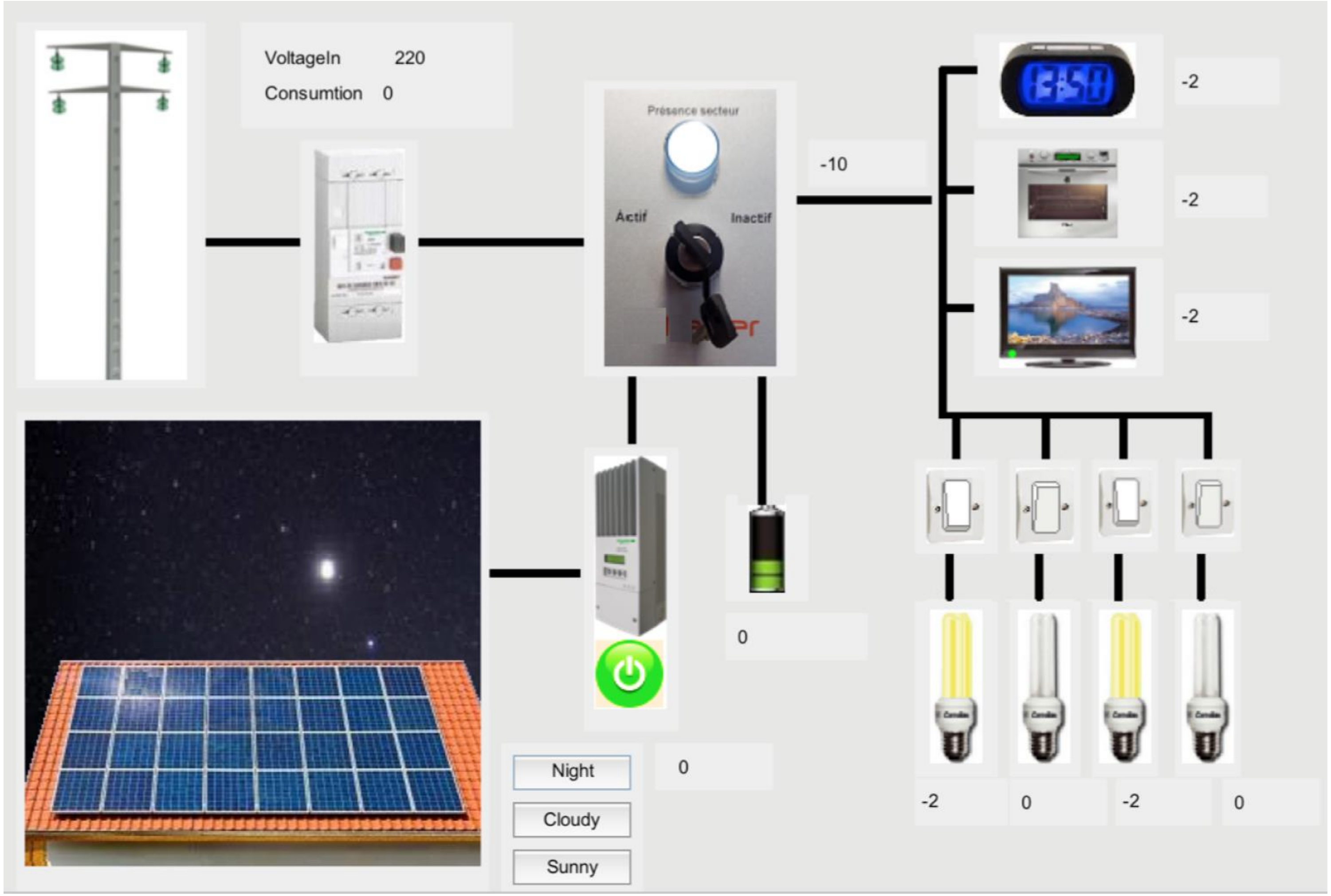
A state machine abstracts system behavior into a finite number of states.

Behaviors are associated with transitions

Behaviors are associated with states

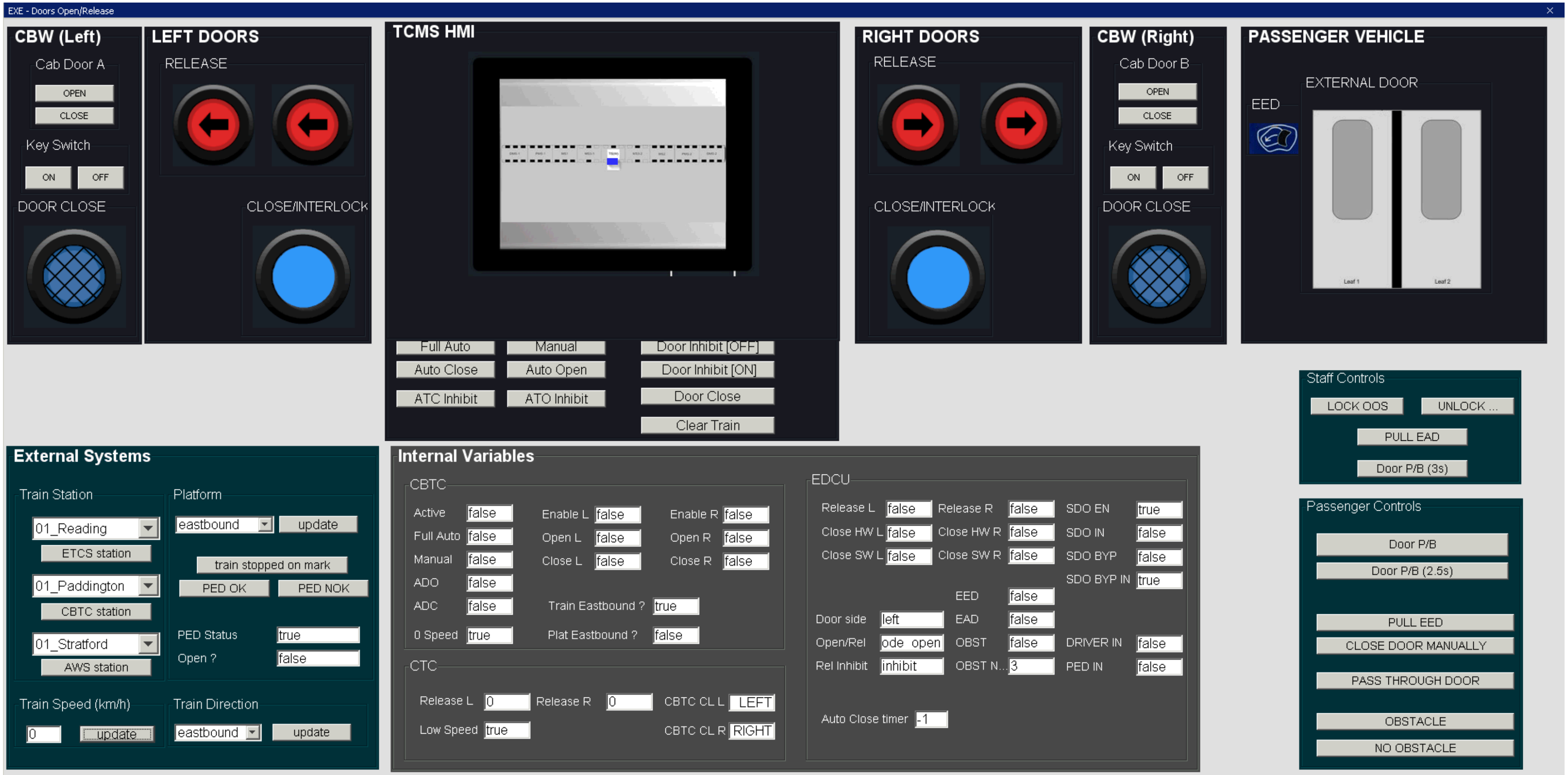
User Interface Mockups

GUI Examples



VoltageIn 220
Consumtion 0

Train doors



External Systems

Train Station

- 01_Reading — ETCS station
- 01_Paddington — CBTC station
- 01_Stratford — AWS station

Platform

- eastbound — update
- train stopped on mark
- PED OK — PED NOK
- PED Status: true
- Open ?: false

Train Speed (km/h): 0 — update

Train Direction: eastbound — update

Internal Variables

CBTC

Active	false	Enable L	false	Enable R	false
Full Auto	false	Open L	false	Open R	false
Manual	false	Close L	false	Close R	false
ADO	false				
ADC	false	Train Eastbound ?	true		
0 Speed	true	Plat Eastbound ?	false		

CTC

Release L	0	Release R	0	CBTC CL L	LEFT
Low Speed	true			CBTC CL R	RIGHT

EDCU

Release L	false	Release R	false	SDO EN	true
Close HW L	false	Close HW R	false	SDO IN	false
Close SW L	false	Close SW R	false	SDO BYP	false
				SDO BYP IN	true
		EED	false		
Door side	left	EAD	false		
Open/Rel	ode open	OBST	false	DRIVER IN	false
Rel Inhibit	inhibit	OBST N...	3	PED IN	false

Auto Close timer: -1

Engineering Analysis

Engineering analysis

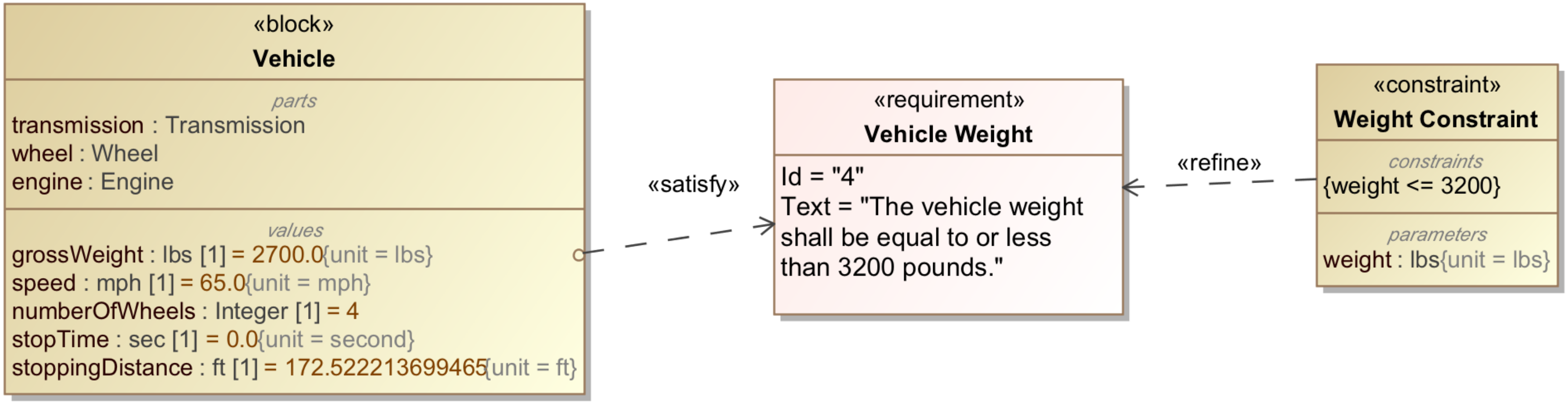
- Automated Requirements Verification
- Trade studies / trade-off analysis
- Total Mass/cost/power rollup
- Model-based testing

Automated Requirements Verification

Formalize requirements

- Represent requirement in BDD
- Identify system parameter which should satisfy the requirement
- Refine requirement into more formal/computable description
- Use constraint block in analysis context
- Bind required system parameters

Step 1. Refine (formalize)



«block»
Vehicle

parts
transmission : Transmission
wheel : Wheel
engine : Engine

values
grossWeight : lbs [1] = 2700.0{unit = lbs}
speed : mph [1] = 65.0{unit = mph}
numberOfWheels : Integer [1] = 4
stopTime : sec [1] = 0.0{unit = second}
stoppingDistance : ft [1] = 172.522213699465{unit = ft}

«satisfy»

«requirement»
Vehicle Weight

Id = "4"
Text = "The vehicle weight shall be equal to or less than 3200 pounds."

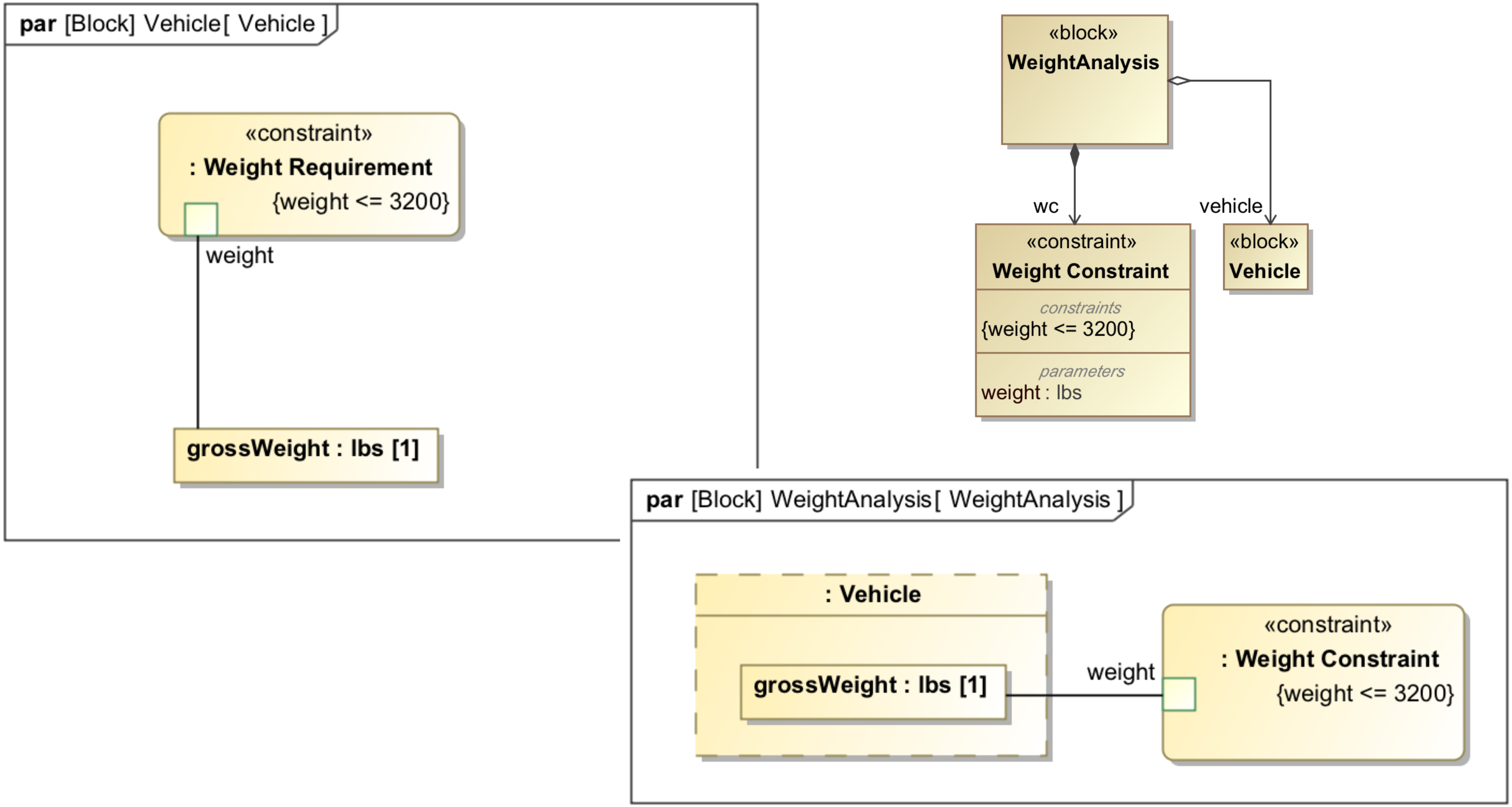
«refine»

«constraint»
Weight Constraint

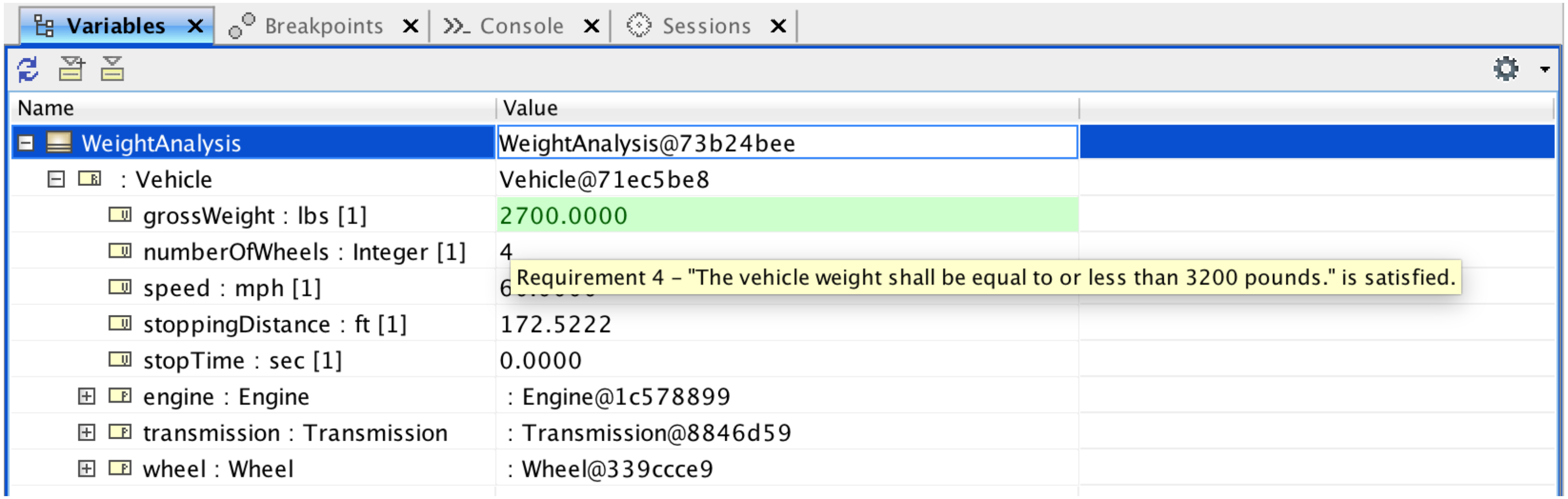
constraints
{weight <= 3200}

parameters
weight : lbs{unit = lbs}

Step 2. Binding

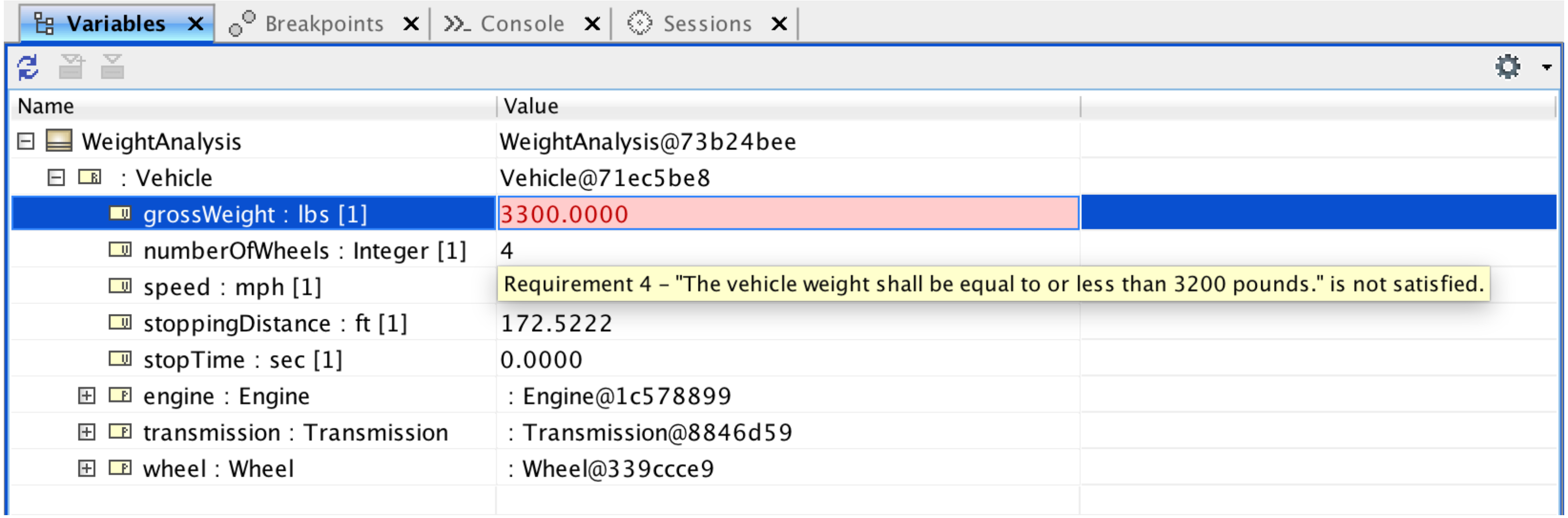


Constraints Verification



Name	Value
WeightAnalysis	WeightAnalysis@73b24bee
: Vehicle	Vehicle@71ec5be8
grossWeight : lbs [1]	2700.0000
numberOfWheels : Integer [1]	4
speed : mph [1]	6
stoppingDistance : ft [1]	172.5222
stopTime : sec [1]	0.0000
engine : Engine	: Engine@1c578899
transmission : Transmission	: Transmission@8846d59
wheel : Wheel	: Wheel@339ccce9

Requirement 4 – "The vehicle weight shall be equal to or less than 3200 pounds." is satisfied.



Name	Value
WeightAnalysis	WeightAnalysis@73b24bee
: Vehicle	Vehicle@71ec5be8
grossWeight : lbs [1]	3300.0000
numberOfWheels : Integer [1]	4
speed : mph [1]	
stoppingDistance : ft [1]	172.5222
stopTime : sec [1]	0.0000
engine : Engine	: Engine@1c578899
transmission : Transmission	: Transmission@8846d59
wheel : Wheel	: Wheel@339ccce9

Requirement 4 – "The vehicle weight shall be equal to or less than 3200 pounds." is not satisfied.

Trade Studies/Trade-Off Analysis

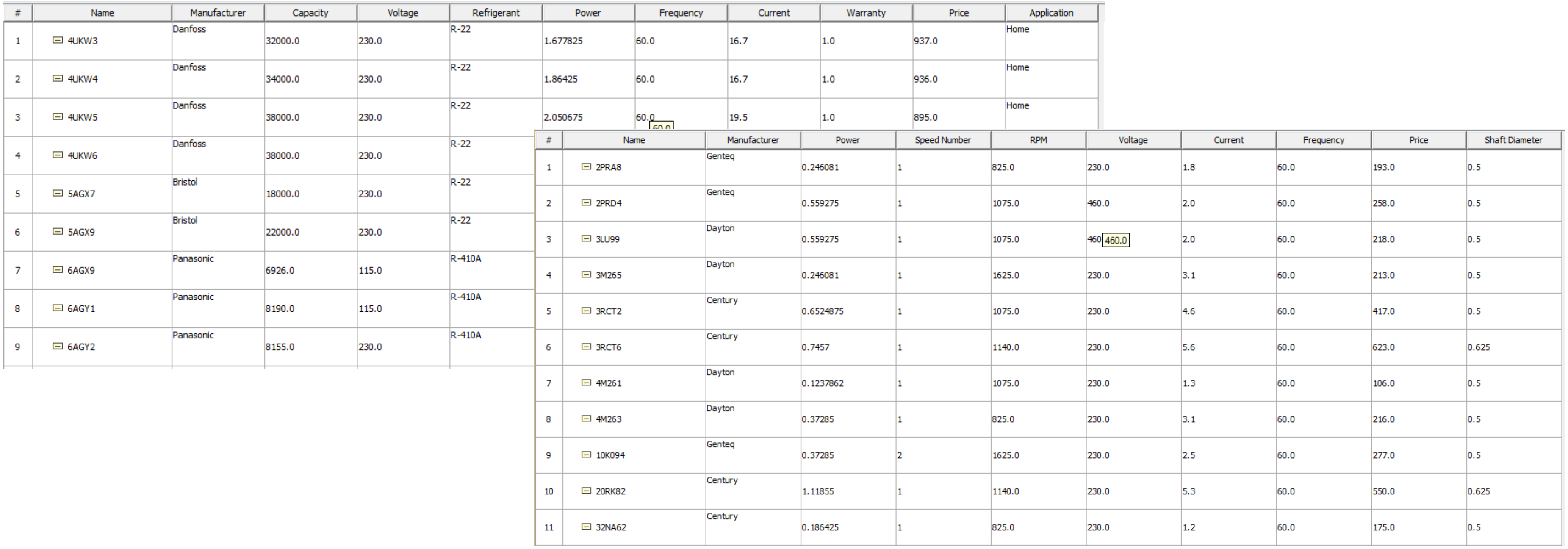
What Is A Trade Study?

A trade study or trade-off study is the activity of a multidisciplinary team to identify the most balanced technical solutions among a set of proposed viable solutions

(System Engineering Manual,
Federal Aviation Administration, 2006)

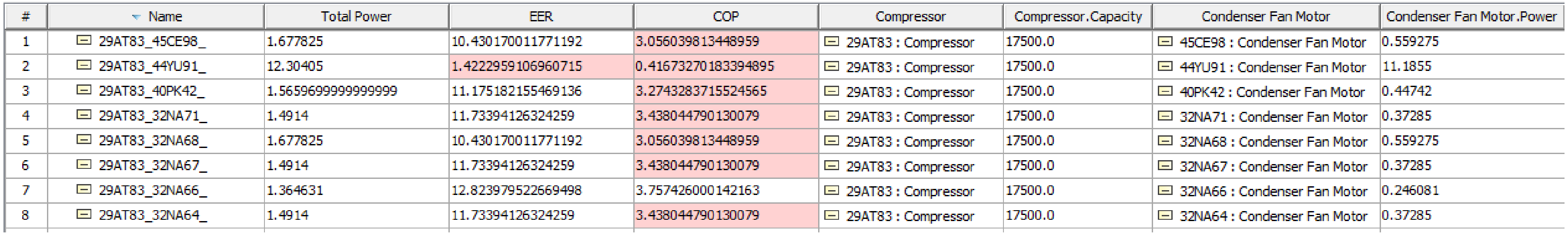
Parametric Trade Study

- Examining various design alternatives by comparison



#	Name	Manufacturer	Capacity	Voltage	Refrigerant	Power	Frequency	Current	Warranty	Price	Application
1	4UKW3	Danfoss	32000.0	230.0	R-22	1.677825	60.0	16.7	1.0	937.0	Home
2	4UKW4	Danfoss	34000.0	230.0	R-22	1.86425	60.0	16.7	1.0	936.0	Home
3	4UKW5	Danfoss	38000.0	230.0	R-22	2.050675	60.0	19.5	1.0	895.0	Home
4	4UKW6	Danfoss	38000.0	230.0	R-22						
5	5AGX7	Bristol	18000.0	230.0	R-22						
6	5AGX9	Bristol	22000.0	230.0	R-22						
7	6AGX9	Panasonic	6926.0	115.0	R-410A						
8	6AGY1	Panasonic	8190.0	115.0	R-410A						
9	6AGY2	Panasonic	8155.0	230.0	R-410A						

#	Name	Manufacturer	Power	Speed Number	RPM	Voltage	Current	Frequency	Price	Shaft Diameter
1	2PRA8	Genteq	0.246081	1	825.0	230.0	1.8	60.0	193.0	0.5
2	2PRD4	Genteq	0.559275	1	1075.0	460.0	2.0	60.0	258.0	0.5
3	3LU99	Dayton	0.559275	1	1075.0	460 460.0	2.0	60.0	218.0	0.5
4	3M265	Dayton	0.246081	1	1625.0	230.0	3.1	60.0	213.0	0.5
5	3RCT2	Century	0.6524875	1	1075.0	230.0	4.6	60.0	417.0	0.5
6	3RCT6	Century	0.7457	1	1140.0	230.0	5.6	60.0	623.0	0.625
7	4M261	Dayton	0.1237862	1	1075.0	230.0	1.3	60.0	106.0	0.5
8	4M263	Dayton	0.37285	1	825.0	230.0	3.1	60.0	216.0	0.5
9	10K094	Genteq	0.37285	2	1625.0	230.0	2.5	60.0	277.0	0.5
10	20RK82	Century	1.11855	1	1140.0	230.0	5.3	60.0	550.0	0.625
11	32NA62	Century	0.186425	1	825.0	230.0	1.2	60.0	175.0	0.5



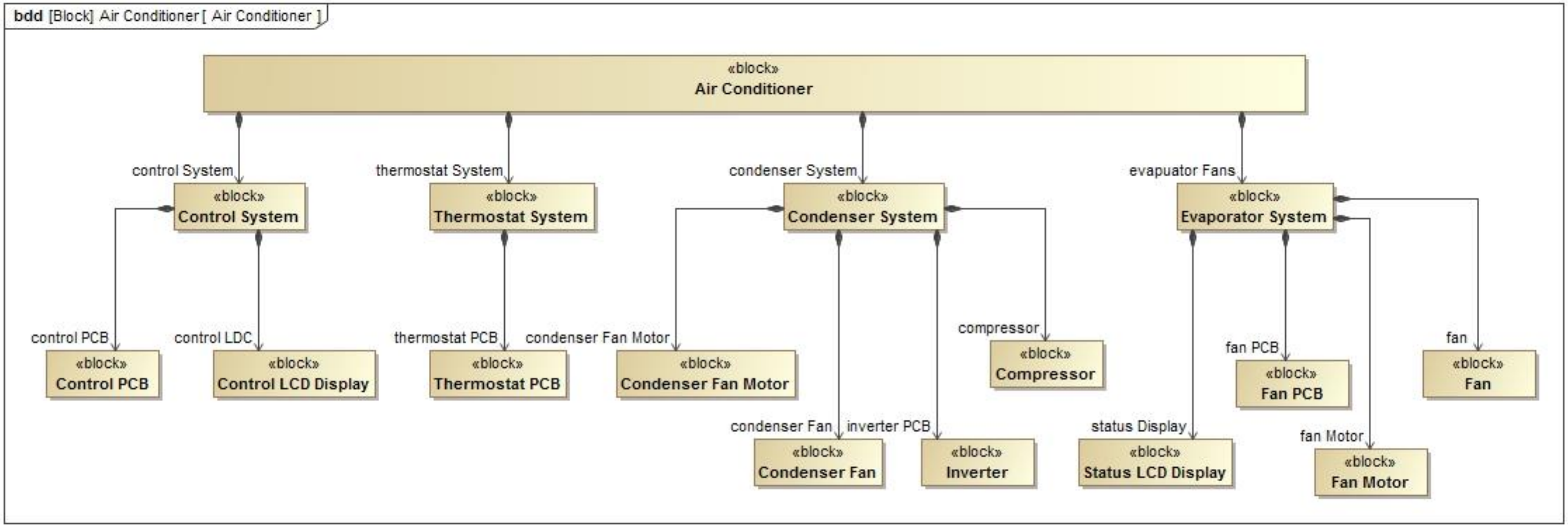
#	Name	Total Power	EER	COP	Compressor	Compressor.Capacity	Condenser Fan Motor	Condenser Fan Motor.Power
1	29AT83_45CE98_	1.677825	10.430170011771192	3.056039813448959	29AT83 : Compressor	17500.0	45CE98 : Condenser Fan Motor	0.559275
2	29AT83_44YU91_	12.30405	1.4222959106960715	0.41673270183394895	29AT83 : Compressor	17500.0	44YU91 : Condenser Fan Motor	11.1855
3	29AT83_40PK42_	1.5659699999999999	11.175182155469136	3.2743283715524565	29AT83 : Compressor	17500.0	40PK42 : Condenser Fan Motor	0.44742
4	29AT83_32NA71_	1.4914	11.73394126324259	3.438044790130079	29AT83 : Compressor	17500.0	32NA71 : Condenser Fan Motor	0.37285
5	29AT83_32NA68_	1.677825	10.430170011771192	3.056039813448959	29AT83 : Compressor	17500.0	32NA68 : Condenser Fan Motor	0.559275
6	29AT83_32NA67_	1.4914	11.73394126324259	3.438044790130079	29AT83 : Compressor	17500.0	32NA67 : Condenser Fan Motor	0.37285
7	29AT83_32NA66_	1.364631	12.823979522669498	3.757426000142163	29AT83 : Compressor	17500.0	32NA66 : Condenser Fan Motor	0.246081
8	29AT83_32NA64_	1.4914	11.73394126324259	3.438044790130079	29AT83 : Compressor	17500.0	32NA64 : Condenser Fan Motor	0.37285

Mass, power, cost rollups

The problem

Requirement:

The total power of air conditioner shall not exceed 1.0 kW.

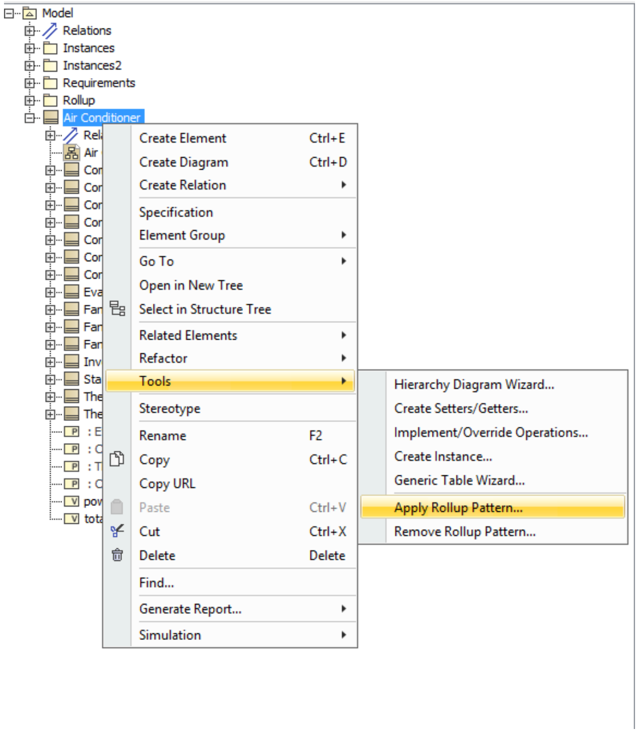


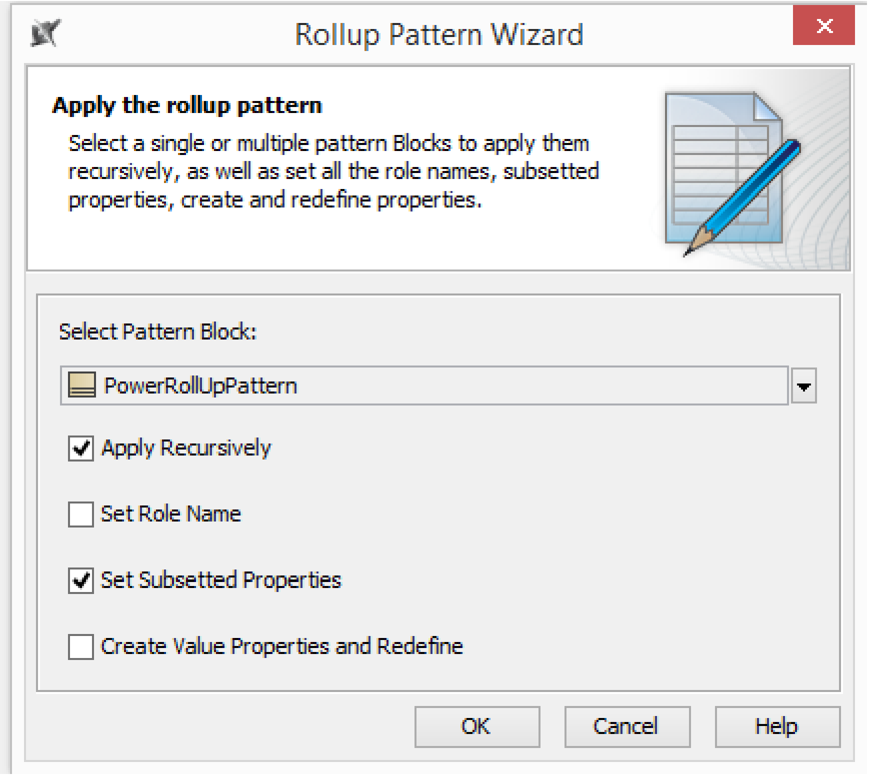
A typical way

- Add property to every block in the structure
- Create Parametric diagrams for every block
- Create “sum(cost1, cost2, …) constraint blocks
- Results
  - many hours of work
  - Polluted model
  - Hundreds of parametric diagrams
  - Need to remodel when add any new part

How could we improve that?

- Use inheritance
  - Define everything in some abstract “CostOwner”
  - Inherit parametrics
- Use subsets – say what roles are children
- Automate the process





Rollup Pattern Wizard

Apply the rollup pattern

Select a single or multiple pattern Blocks to apply them recursively, as well as set all the role names, subsetted properties, create and redefine properties.

Select Pattern Block: PowerRollUpPattern

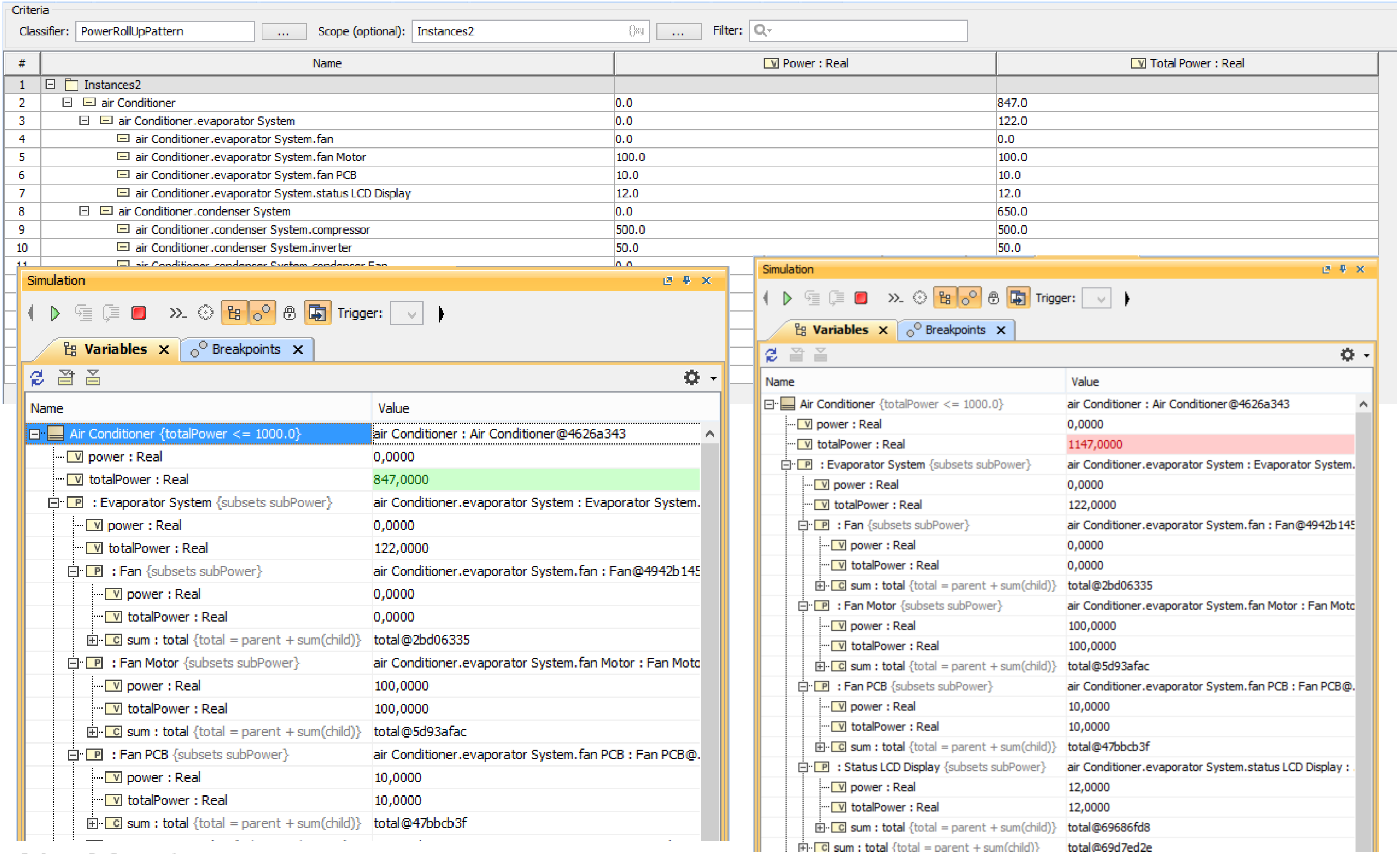
☑ Apply Recursively

□ Set Role Name

☑ Set Subsetted Properties

□ Create Value Properties and Redefine

Run Analysis



Criteria

Classifier: PowerRollUpPattern ... Scope (optional): Instances2 ... Filter:

#	Name	Power : Real	Total Power : Real
1	Instances2		
2	air Conditioner	0.0	847.0
3	air Conditioner.evaporator System	0.0	122.0
4	air Conditioner.evaporator System.fan	0.0	0.0
5	air Conditioner.evaporator System.fan Motor	100.0	100.0
6	air Conditioner.evaporator System.fan PCB	10.0	10.0
7	air Conditioner.evaporator System.status LCD Display	12.0	12.0
8	air Conditioner.condenser System	0.0	650.0
9	air Conditioner.condenser System.compressor	500.0	500.0
10	air Conditioner.condenser System.inverter	50.0	50.0
11	air Conditioner.condenser System.condenser Fan	0.0	

Simulation

Trigger:

Variables | Breakpoints

Name	Value
Air Conditioner {totalPower <= 1000.0}	air Conditioner : Air Conditioner@4626a343
power : Real	0,0000
totalPower : Real	847,0000
: Evaporator System {subsets subPower}	air Conditioner.evaporator System : Evaporator System.
power : Real	0,0000
totalPower : Real	122,0000
: Fan {subsets subPower}	air Conditioner.evaporator System.fan : Fan@4942b145
power : Real	0,0000
totalPower : Real	0,0000
sum : total {total = parent + sum(child)}	total@2bd06335
: Fan Motor {subsets subPower}	air Conditioner.evaporator System.fan Motor : Fan Moto
power : Real	100,0000
totalPower : Real	100,0000
sum : total {total = parent + sum(child)}	total@5d93afac
: Fan PCB {subsets subPower}	air Conditioner.evaporator System.fan PCB : Fan PCB@.
power : Real	10,0000
totalPower : Real	10,0000
sum : total {total = parent + sum(child)}	total@47bbcb3f

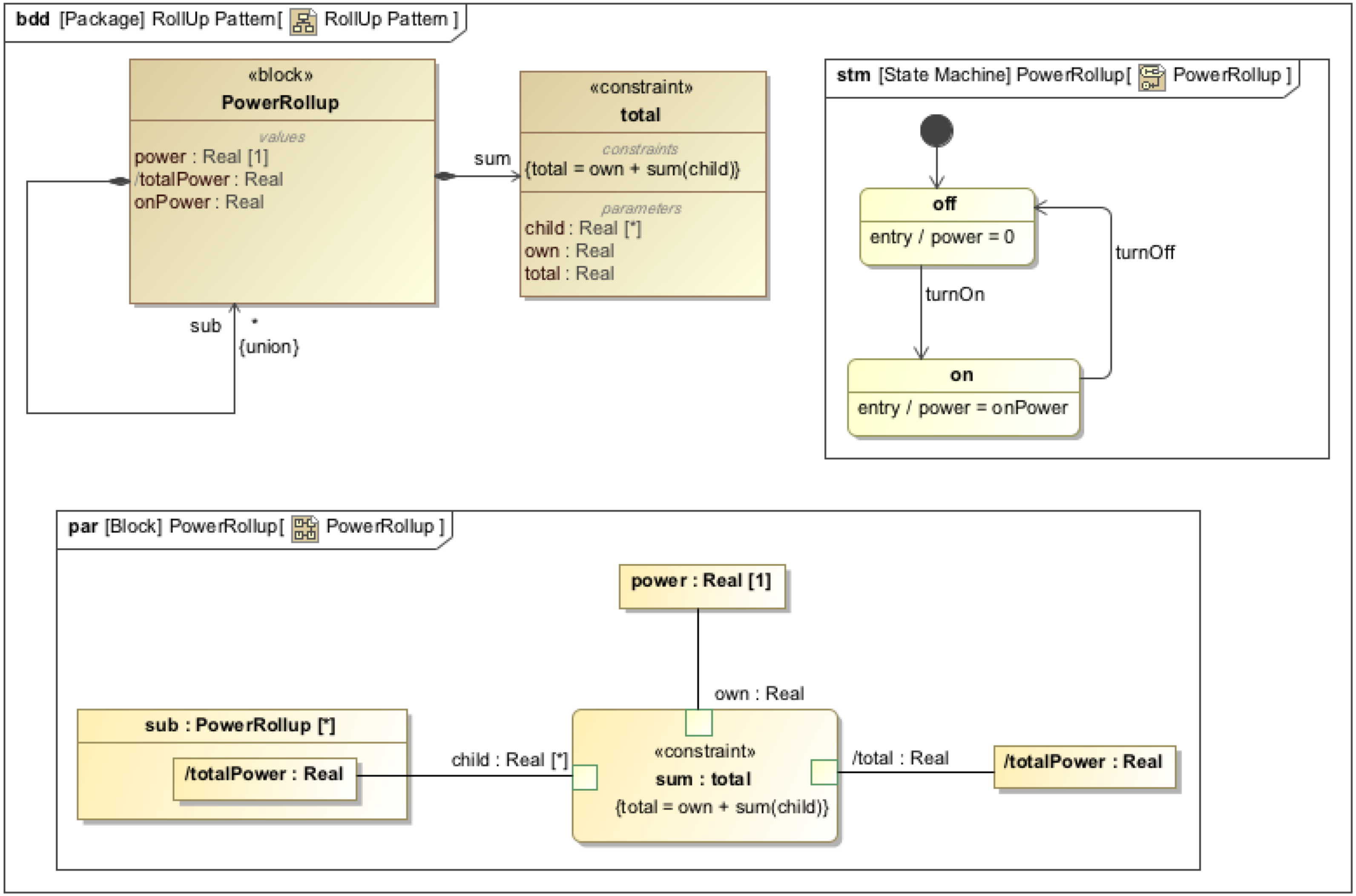
Simulation

Trigger:

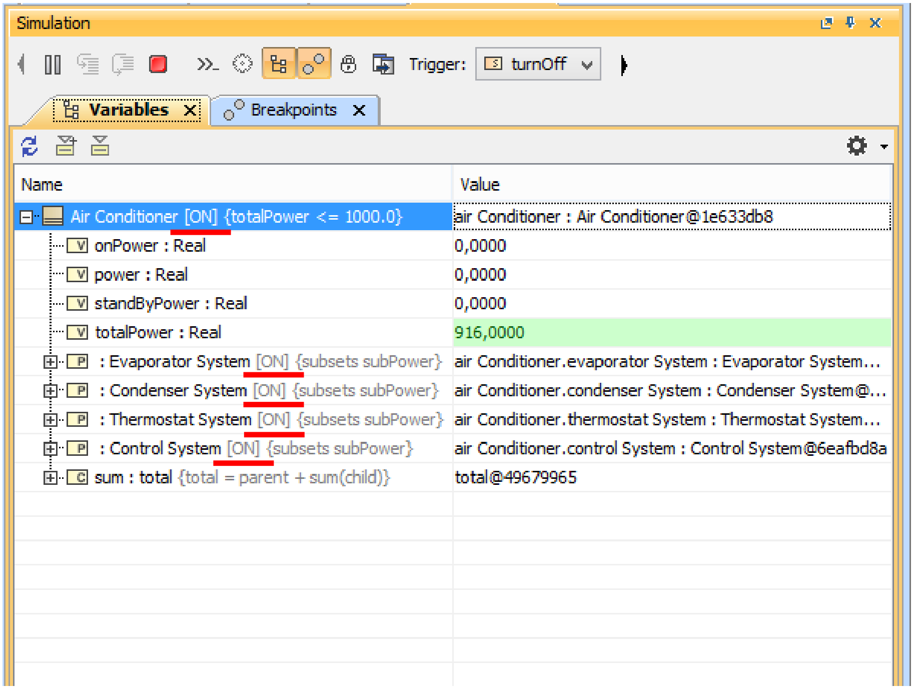
Variables | Breakpoints

Name	Value
Air Conditioner {totalPower <= 1000.0}	air Conditioner : Air Conditioner@4626a343
power : Real	0,0000
totalPower : Real	1147,0000
: Evaporator System {subsets subPower}	air Conditioner.evaporator System : Evaporator System.
power : Real	0,0000
totalPower : Real	122,0000
: Fan {subsets subPower}	air Conditioner.evaporator System.fan : Fan@4942b145
power : Real	0,0000
totalPower : Real	0,0000
sum : total {total = parent + sum(child)}	total@2bd06335
: Fan Motor {subsets subPower}	air Conditioner.evaporator System.fan Motor : Fan Moto
power : Real	100,0000
totalPower : Real	100,0000
sum : total {total = parent + sum(child)}	total@5d93afac
: Fan PCB {subsets subPower}	air Conditioner.evaporator System.fan PCB : Fan PCB@.
power : Real	10,0000
totalPower : Real	10,0000
sum : total {total = parent + sum(child)}	total@47bbcb3f
: Status LCD Display {subsets subPower}	air Conditioner.evaporator System.status LCD Display :
power : Real	12,0000
totalPower : Real	12,0000
sum : total {total = parent + sum(child)}	total@69686fd8
sum : total {total = parent + sum(child)}	total@69d7ed2e

Dynamic rollup



Dynamic rollup

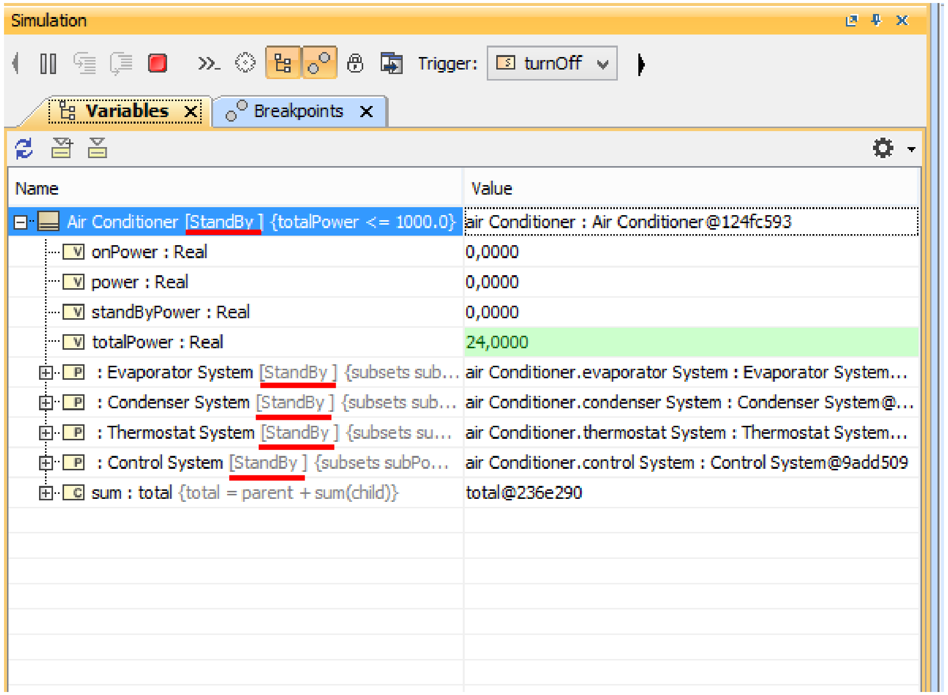


Simulation

Trigger: turnOff

Variables | Breakpoints

Name	Value
Air Conditioner [ON] {totalPower <= 1000.0}	air Conditioner : Air Conditioner@1e633db8
onPower : Real	0,0000
power : Real	0,0000
standByPower : Real	0,0000
totalPower : Real	916,0000
: Evaporator System [ON] {subsets subPower}	air Conditioner.evaporator System : Evaporator System...
: Condenser System [ON] {subsets subPower}	air Conditioner.condenser System : Condenser System@...
: Thermostat System [ON] {subsets subPower}	air Conditioner.thermostat System : Thermostat System...
: Control System [ON] {subsets subPower}	air Conditioner.control System : Control System@6eafbd8a
sum : total {total = parent + sum(child)}	total@49679965



Simulation

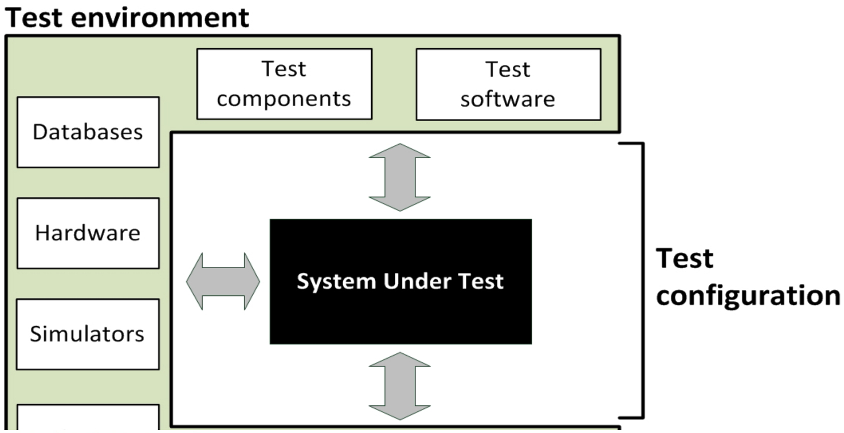
Trigger: turnOff

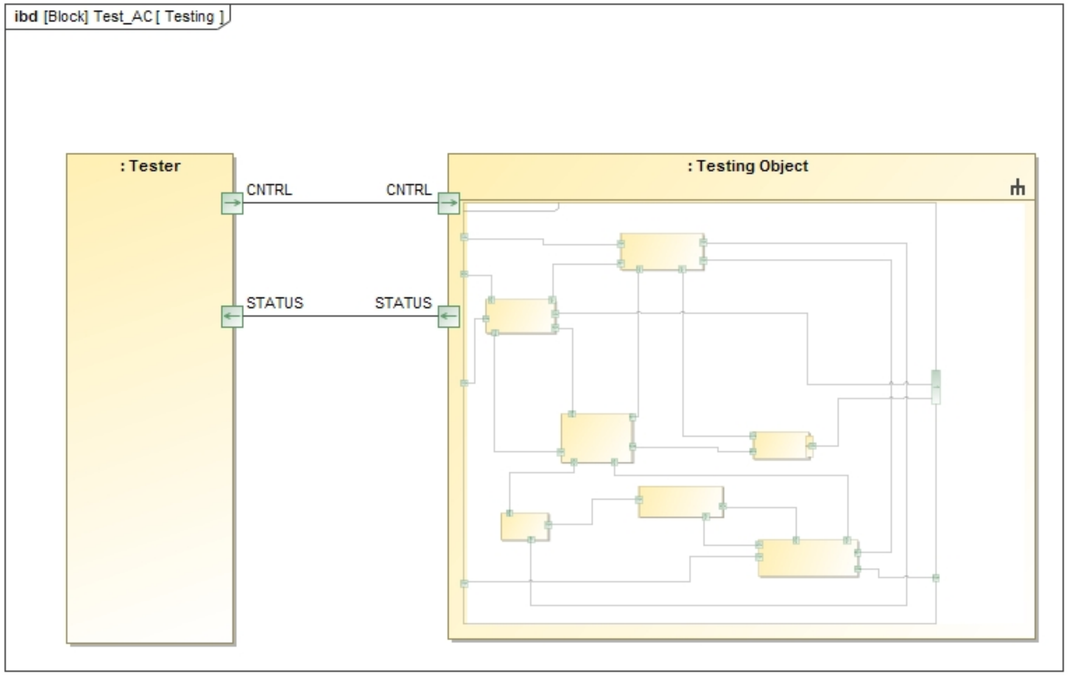
Variables | Breakpoints

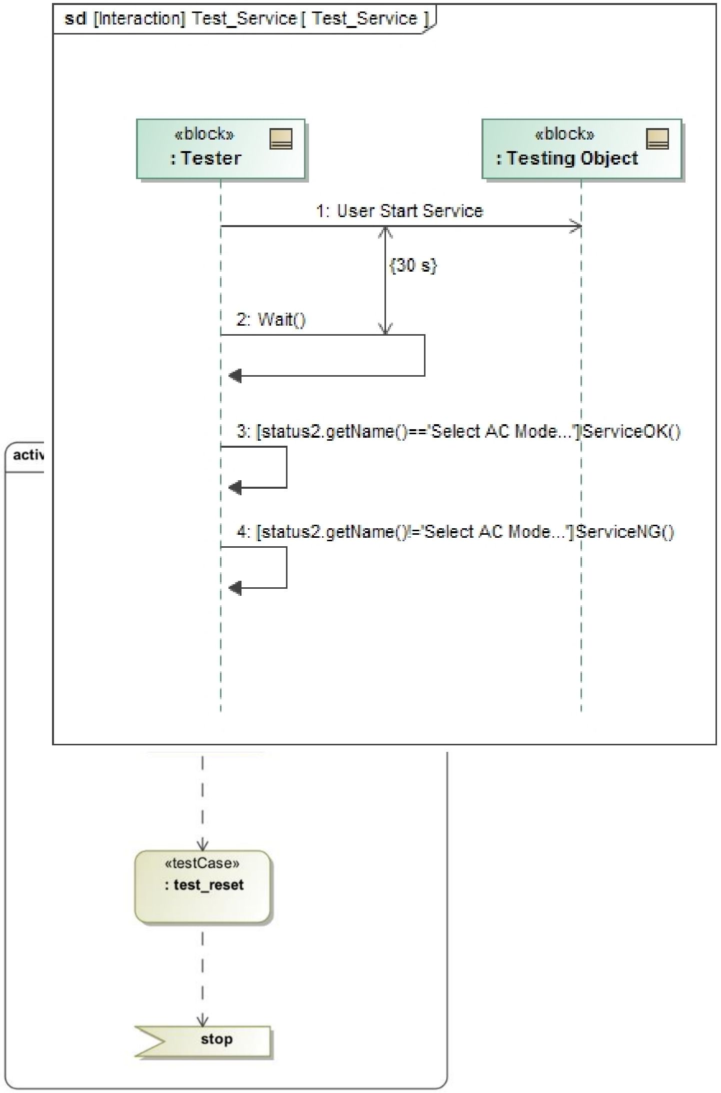
Name	Value
Air Conditioner [StandBy] {totalPower <= 1000.0}	air Conditioner : Air Conditioner@124fc593
onPower : Real	0,0000
power : Real	0,0000
standByPower : Real	0,0000
totalPower : Real	24,0000
: Evaporator System [StandBy] {subsets sub...	air Conditioner.evaporator System : Evaporator System...
: Condenser System [StandBy] {subsets sub...	air Conditioner.condenser System : Condenser System@...
: Thermostat System [StandBy] {subsets su...	air Conditioner.thermostat System : Thermostat System...
: Control System [StandBy] {subsets subPo...	air Conditioner.control System : Control System@9add509
sum : total {total = parent + sum(child)}	total@236e290

Model based testing

Model-based testing in SysML

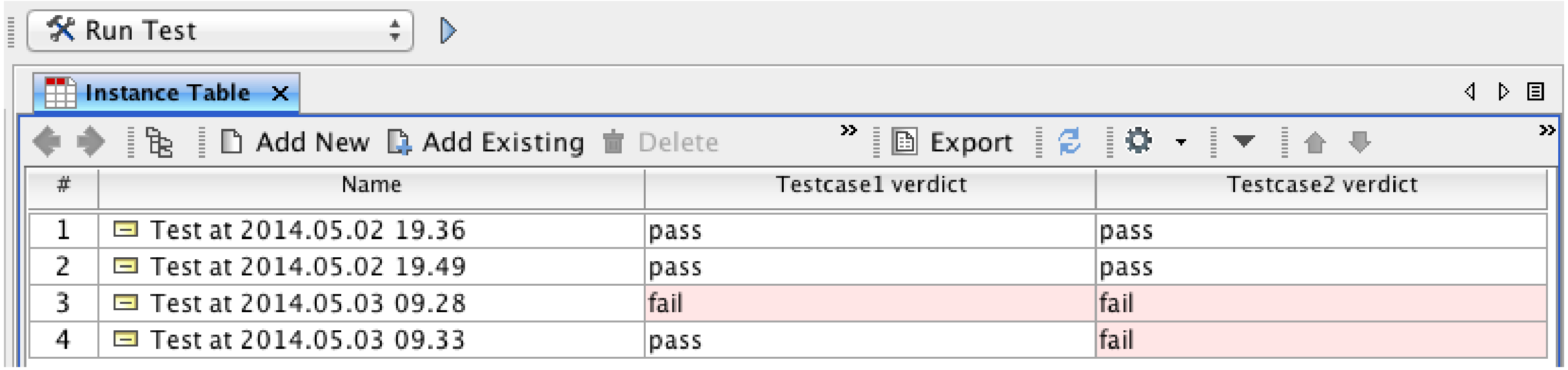




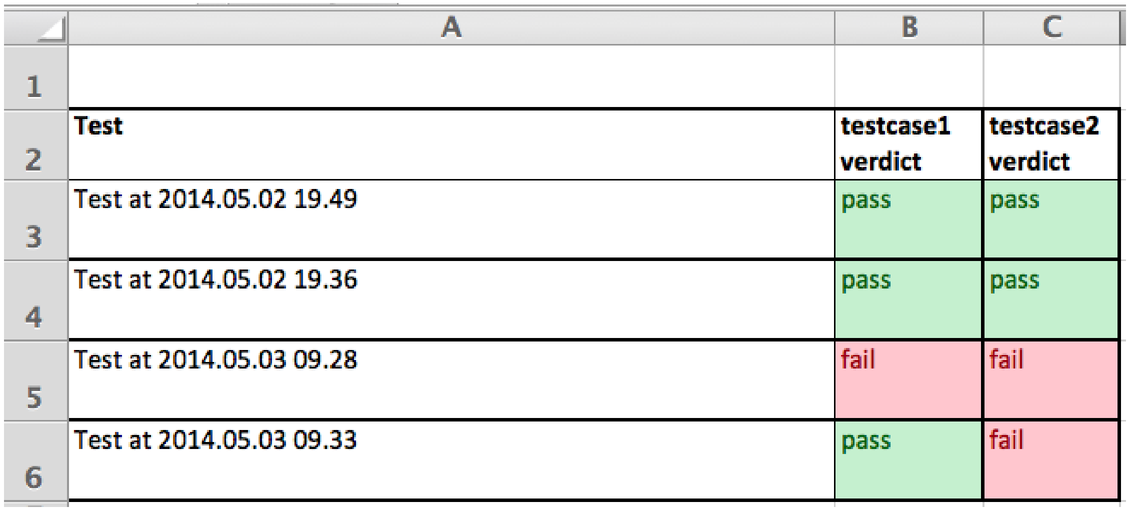


sd [Interaction] Test_Service [Test_Service]
«block»
: Tester
«block»
: Testing Object
1: User Start Service
{30 s}
2: Wait()
3: [status2.getName()=='Select AC Mode...']ServiceOK()
4: [status2.getName()!='Select AC Mode...']ServiceNG()
activ
«testCase»
: test_reset
stop

Testing report (Instance Table + Excel)



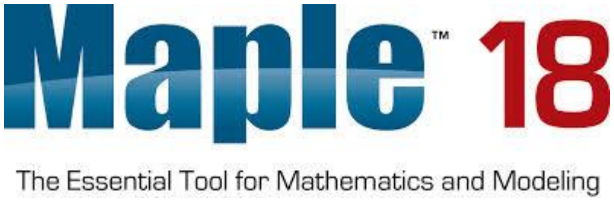
#	Name	Testcase1 verdict	Testcase2 verdict
1	Test at 2014.05.02 19.36	pass	pass
2	Test at 2014.05.02 19.49	pass	pass
3	Test at 2014.05.03 09.28	fail	fail
4	Test at 2014.05.03 09.33	pass	fail



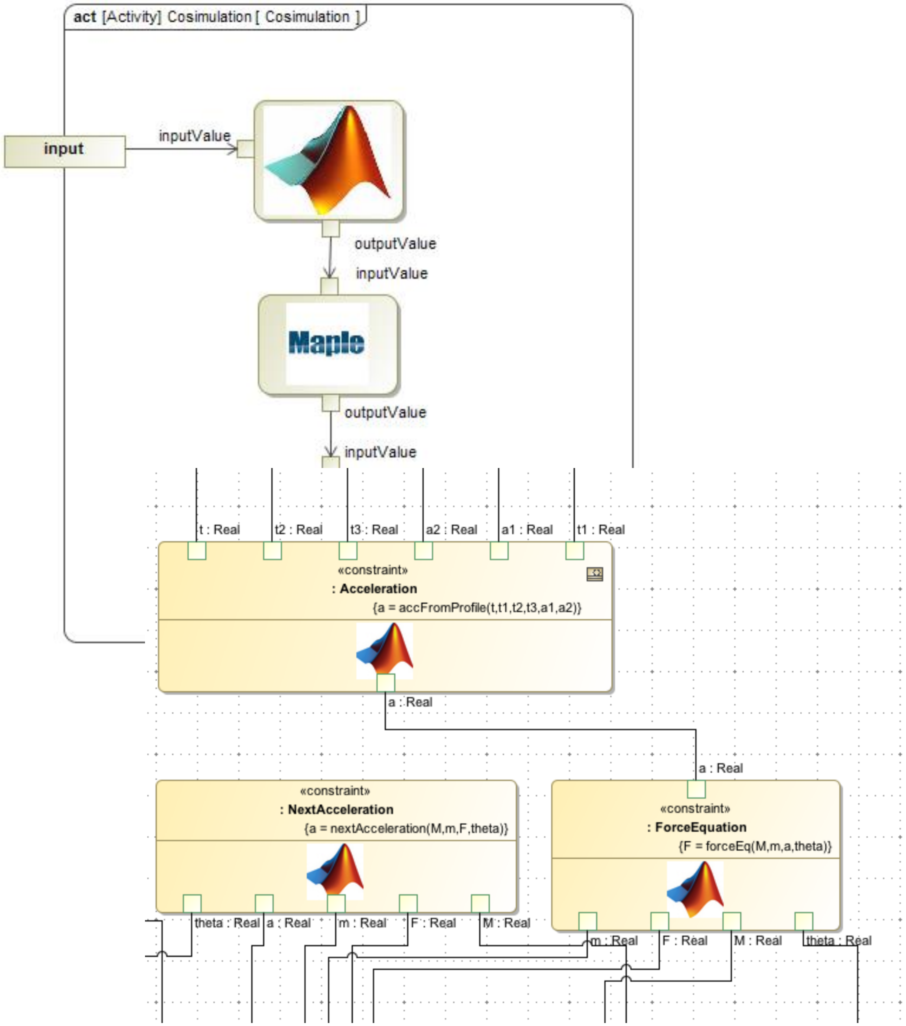
	A	B	C
1			
2	**Test**	**testcase1 verdict**	**testcase2 verdict**
3	Test at 2014.05.02 19.49	pass	pass
4	Test at 2014.05.02 19.36	pass	pass
5	Test at 2014.05.03 09.28	fail	fail
6	Test at 2014.05.03 09.33	pass	fail

Integration of Analytics Model

Integration of Analytics Models



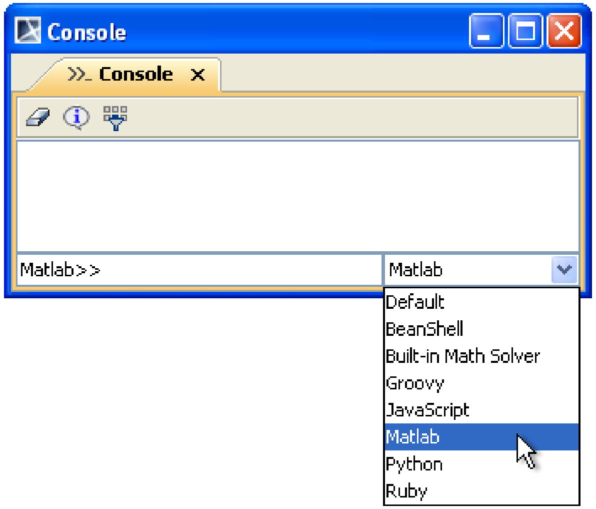
Cameo Simulation Toolkit



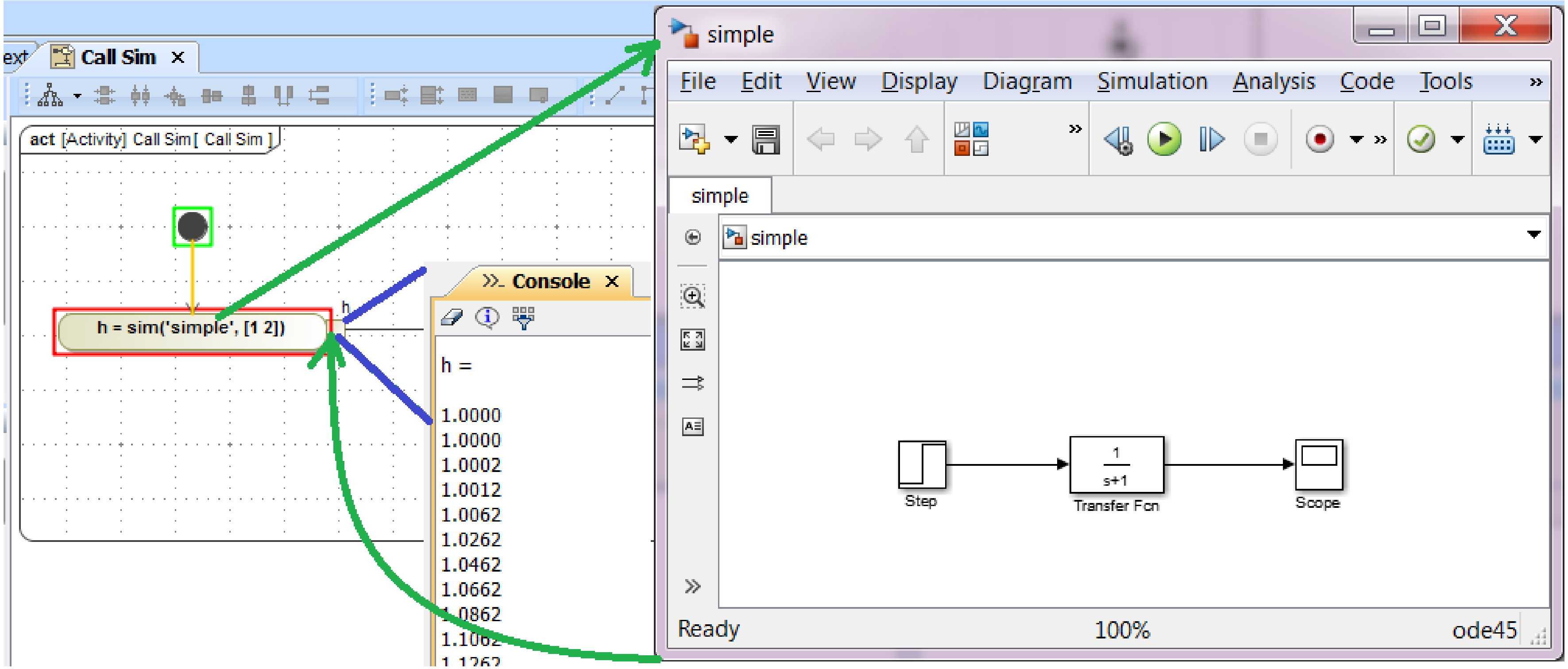
Math engines

- Matlab/Simulink
- Mathematica
- Maple
- Open Modelica

- Scripting
  - Javascript
  - Python
  - Groovy
  - Ruby



Co-simulation: Invoking Simulink Model

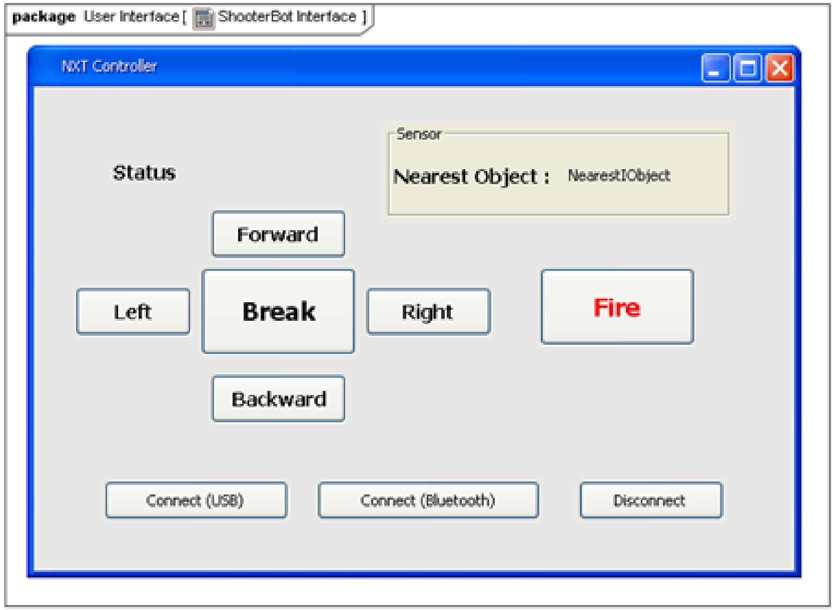


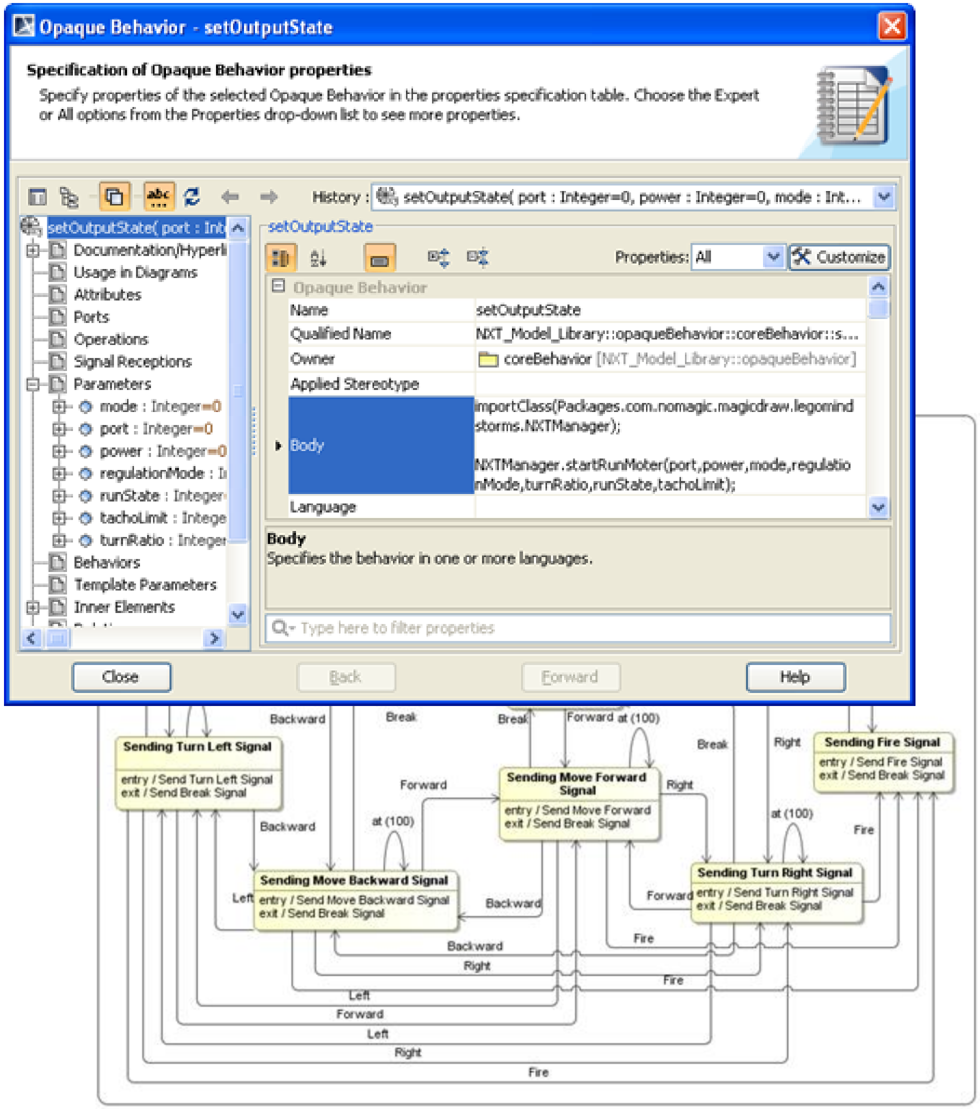
```
h = sim('simple', [1 2])
```

```
h =

1.0000
1.0000
1.0002
1.0012
1.0062
1.0262
1.0462
1.0662
1.0862
1.1062
1.1262
```

Scripting, external APIs





Opaque Behavior - setOutputState

Specification of Opaque Behavior properties

Specify properties of the selected Opaque Behavior in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

History : setOutputState(port : Integer=0, power : Integer=0, mode : Int...

Properties: All

Opaque Behavior	
Name	setOutputState
Qualified Name	NXT_Model_Library::opaqueBehavior::coreBehavior::s...
Owner	coreBehavior [NXT_Model_Library::opaqueBehavior]
Applied Stereotype	

Body	importClass(Packages.com.nomagic.magicdraw.legomindstorms.NXTManager);
	
	NXTManager.startRunMoter(port,power,mode,regulationMode,turnRatio,runState,tachoLimit);

Language	

Body

Specifies the behavior in one or more languages.

Summary

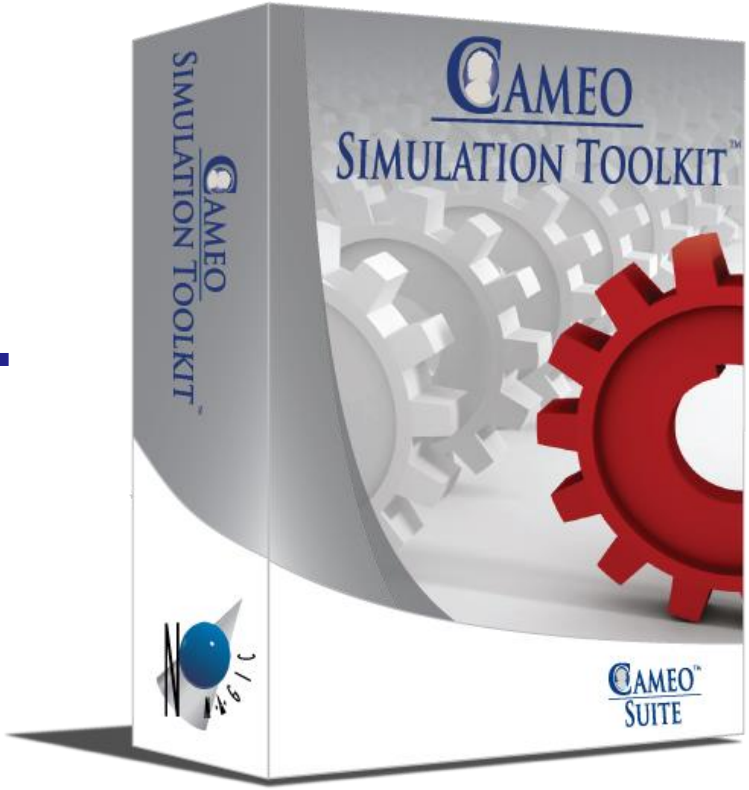
With simulation you can:

- Enhance user understanding by using animation and debugging
- Define your system behavior by using activity and states diagrams
- Create user interface mockup
- Do engineering analysis by using rollups, trade studies, system testing
- Integrate with other analytic tools

Thank You !

CAMEO
SYSTEMS MODELER™
CAMEO™
SUITE

+


CAMEO
SIMULATION TOOLKIT™
CAMEO
SIMULATION TOOLKIT™
NO MAGIC
CAMEO™
SUITE