26th annual INCOSE international symposium

Edinburgh, UK
July 18 - 21, 2016

Towards the Automation of Model-Based Design Verification

Wladimir Schamai, Nicolas Albarello, and Philipp Helle
(Airbus Group Innovations, Germany and France)

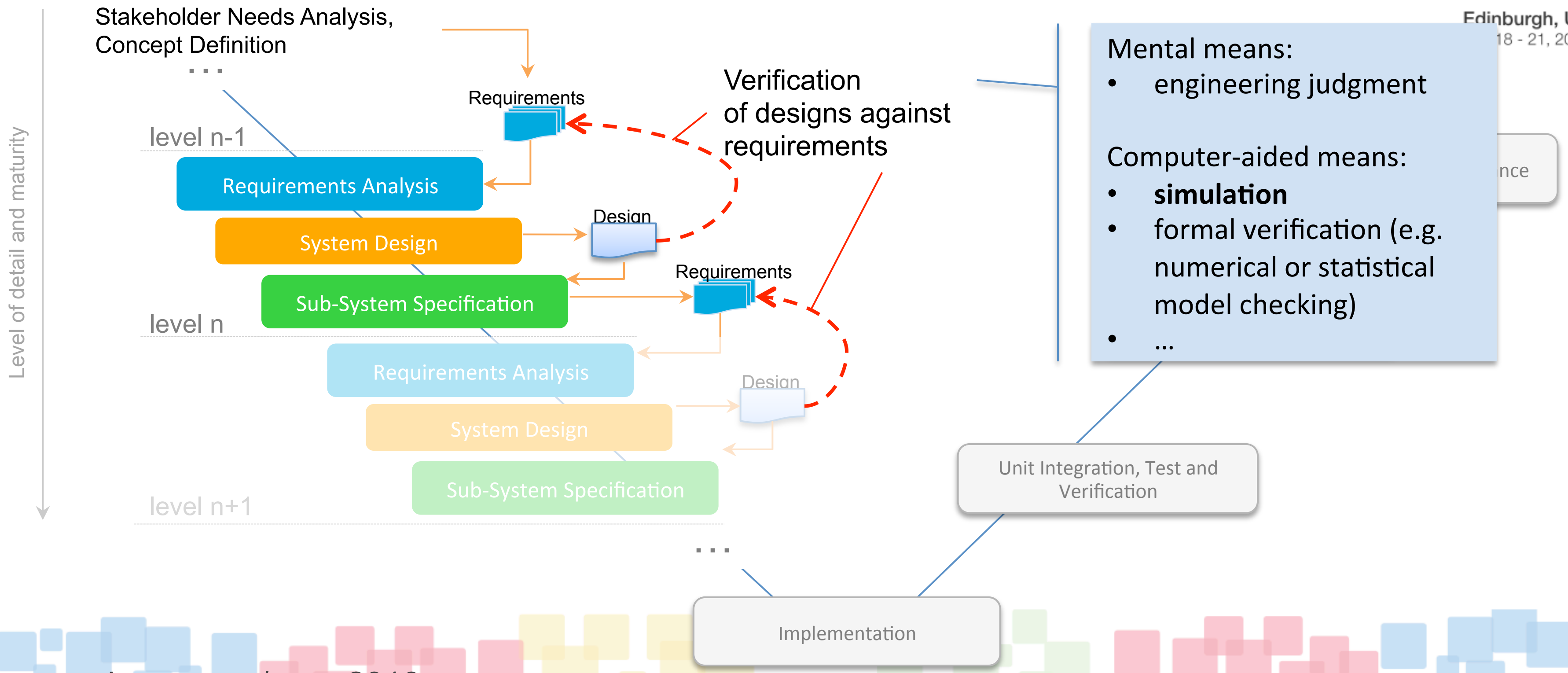
Lena Buffoni and Peter Fritzson
(IDA, Linköping University, Sweden)

Agenda

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

- Introduction
- Method Model-Based Design Verification
- Approach for Requirements Formalization
- Approach for Automated Simulation Model Composition
- Conclusion

Introduction



Mental means:

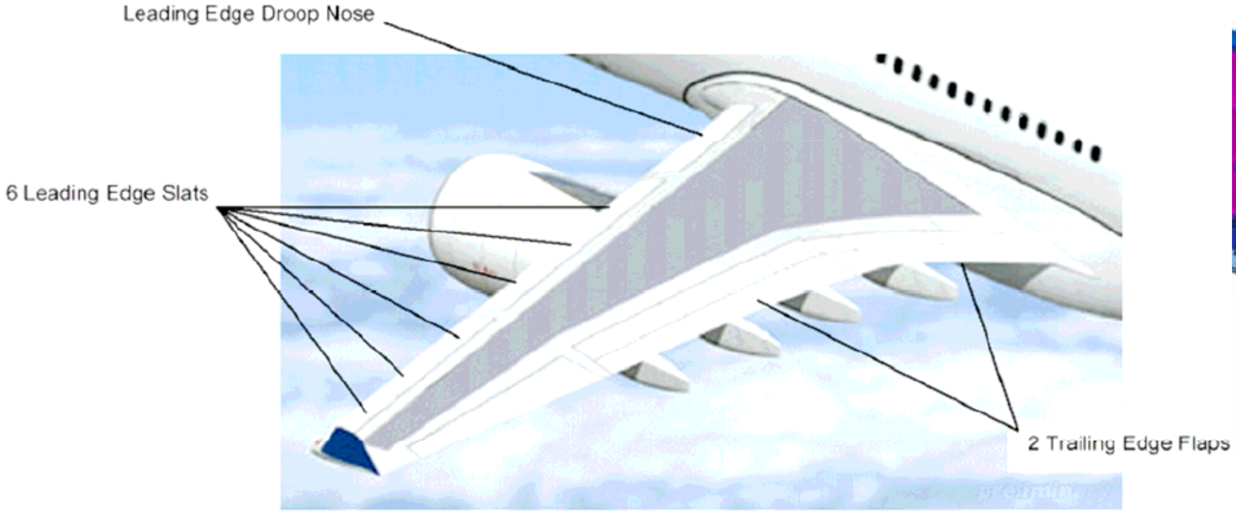
- engineering judgment

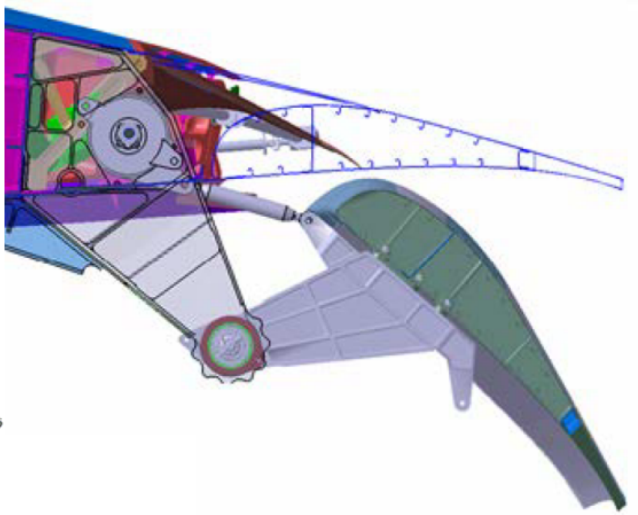
Computer-aided means:

- **simulation**
- formal verification (e.g. numerical or statistical model checking)
- …

Example

- Secondary flight control system (SFCS) allows modifying the wing geometry
- System is composed of spoilers, flaps, slats, electrical motors and actuators on each wing



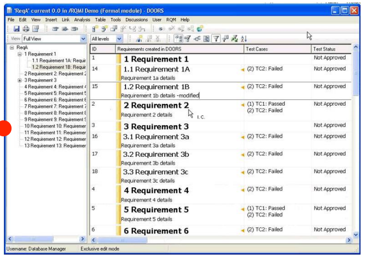


Example

- “The torque of any electrical motor shall not be greater than 20 Nm for more than 1 sec.”
- “The time of any action of flaps actuation (extension/retraction) shall be less than 50 sec.”.
- “The flap angle shall not exceed the range [-5°;35°]”.
- “The force between a flap and its spoiler shall be less than 1000N”.

Introduction

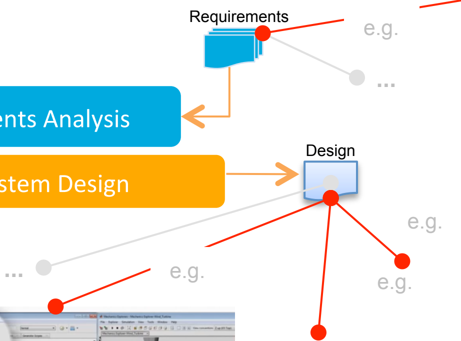
IBM DOORS



'ReqA' current 0.0 in /RQMI Demo (Formal module) - DOORS

ID	Requirements created in DOORS	Test Cases	Test Status
1	1 Requirement 1		Not Approved
14	1.1 Requirement 1A Requirement 1a details	(2) TC2: Failed	Not Approved
15	1.2 Requirement 1B Requirement 1b details -modified	(2) TC2: Failed	Not Approved
2	2 Requirement 2 Requirement 2 details	(1) TC1: Passed (2) TC2: Failed	Not Approved
3	3 Requirement 3		Not Approved
16	3.1 Requirement 3a Requirement 3a details	(2) TC2: Failed	Not Approved
17	3.2 Requirement 3b Requirement 3b details	(2) TC2: Failed	Not Approved
18	3.3 Requirement 3c Requirement 3c details	(2) TC2: Failed	Not Approved
4	4 Requirement 4 Requirement 4 details	(2) TC2: Failed	Not Approved
5	5 Requirement 5 Requirement 5 details	(1) TC1: Passed (2) TC2: Failed	Not Approved
6	6 Requirement 6	(2) TC2: Failed	Not Approved

Exclusive edit mode



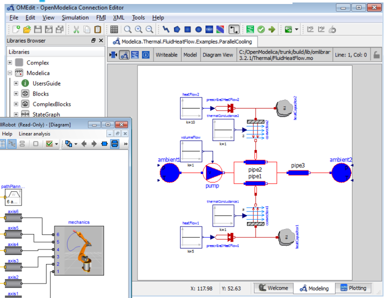
Requirements Analysis

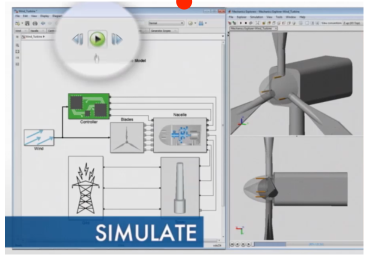
System Design

Natural-language statements ☹

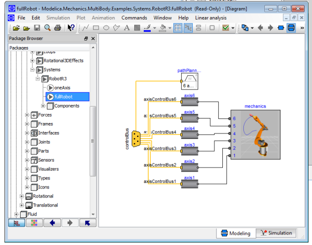
Model-based verification of designs against requirements

OpenModelica





MathWorks Simulink

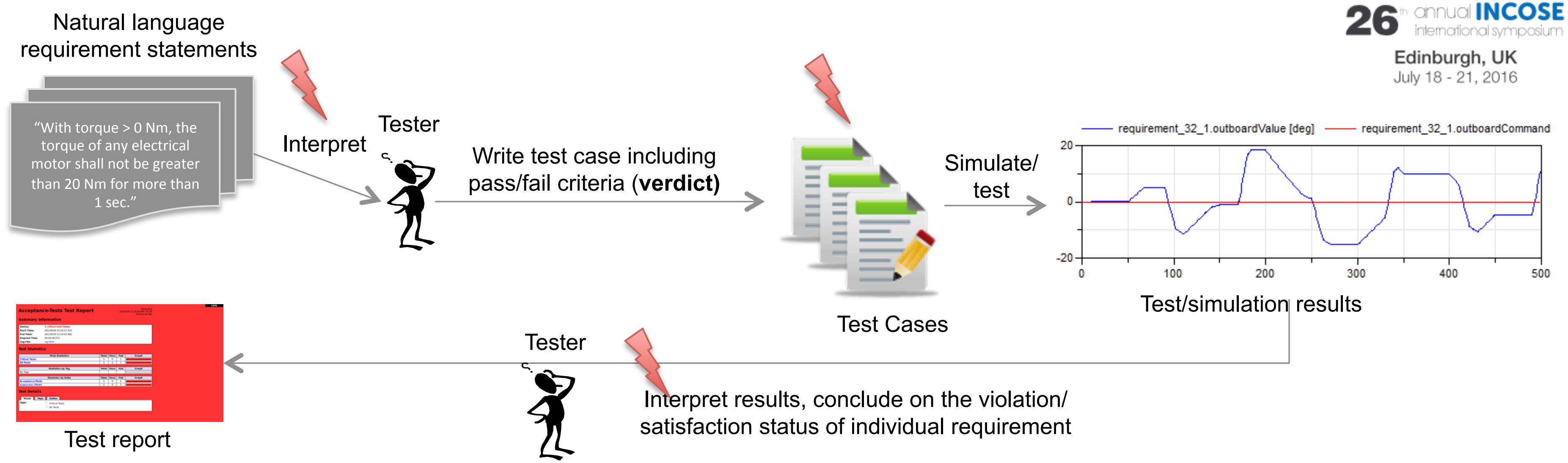

fullRobot - Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot (Read-Only) - [Diagram]
Package Browser
pathPlanning
mechanics
Modeling
Simulation

DASSAULT SYSTEMES Dymola

Executable design models ☺

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Traditional way


Natural language requirement statements
Interpret
Tester
Write test case including pass/fail criteria (**verdict**)
Test Cases
Simulate/ test
requirement_32_1.outboardValue [deg]
requirement_32_1.outboardCommand
20
0
-20
0
100
200
300
400
500
Test/simulation results
Tester
Test report

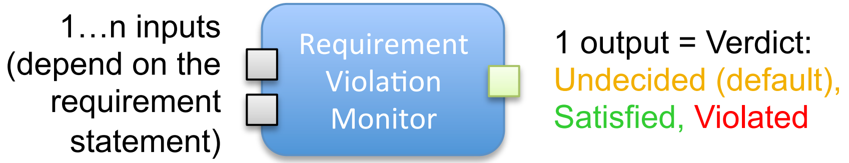
"With torque > 0 Nm, the torque of any electrical motor shall not be greater than 20 Nm for more than 1 sec."

Interpret results, conclude on the violation/ satisfaction status of individual requirement

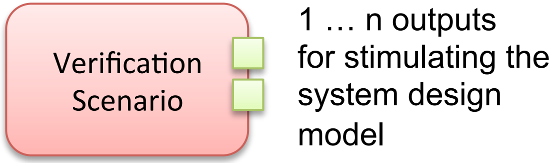
- Interpreting NL statements and simulation results is an error-prone and tedious tasks
- Test case pass/fail depend/focus on requirements
- Test case pass/fail result requires interpretation for concluding on individual req. status

New approach

- Each requirement is formalized into a violation monitor model that detects requirement violations at any simulated time

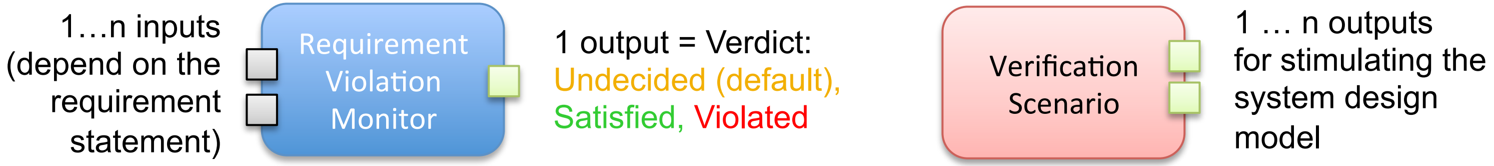

1…n inputs (depend on the requirement statement)
Requirement Violation Monitor
1 output = Verdict:
Undecided (default),
Satisfied, Violated

- Verification scenario model has no pass/fail verdict



New approach

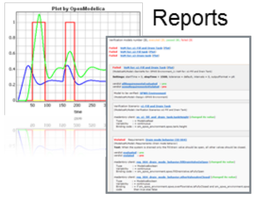
- **Re-use**: Requirement violation monitor models
  - Can be used for verifying any relevant design alternative/version using any appropriate scenario, either in **simulations** or **HiL testing**
  - Verdict output enables **automatic conclusion** on verification status
- **Focus**: Scenarios focus on covering relevant **operational situations** (e.g., normal operation, degraded mode, stress situations, etc.)
- **Coverage**: Same scenarios will be used for testing multiple requirements - requirements will be tested using different scenarios


1…n inputs (depend on the requirement statement)
Requirement Violation Monitor
1 output = Verdict: Undecided (default), Satisfied, Violated
Verification Scenario
1 … n outputs for stimulating the system design model

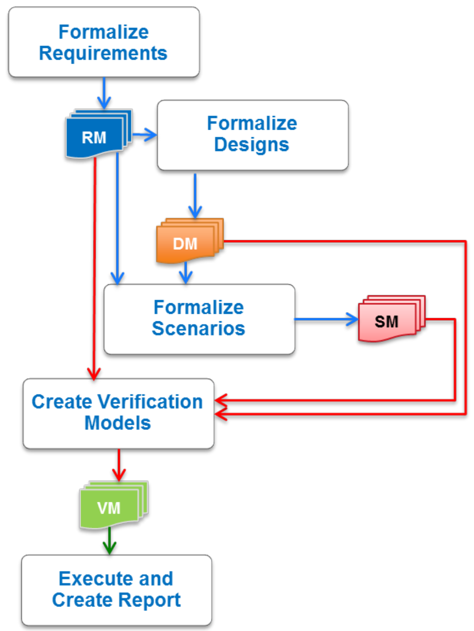
Model-Based Design Verification Method

virtual Verification of Designs against Requirements (vVDR)

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Role	Task	Created Artifact
Requirement Analyst	Formalize Requirements	RM Requirement Models
System Designer	Formalize Designs	DM Design Models
Tester	Formalize Scenarios	SM Scenario Models
Tester, all	Create Verification Models	VM Verification Models
Tester	Execute and Create Report	Reports 
System Designer, all	Analyze Results	

Data Flow



REQUIREMENT FORMALIZATION

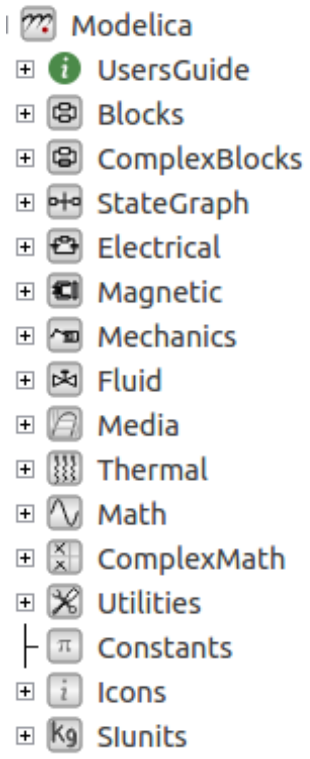
Modelica Introduction

Goal of **Modelica**:

- Modeling the **dynamic behavior** of **technical systems** consisting of components from, e.g., mechanical, electrical, thermal, hydraulic, pneumatic, fluid, control and other domains in a **convenient way**.
- Models are described by **differential, algebraic**, and **discrete equations**.
- No description by partial differential equations, i.e.,
  no FEM (finite element method) and
  no CFD (computational fluid dynamics),
  but using results of, e.g., FEM programs.
- Modelica is used in industry since year 2000.

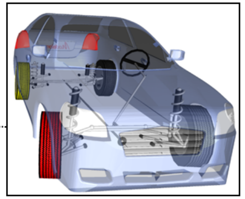
Adapted from: Modelica Overview, Martin Otter, se www.modelica.org

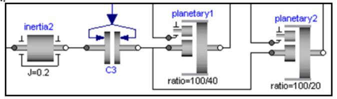
Modelica libraries

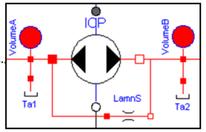

Modelica
- UsersGuide
- Blocks
- ComplexBlocks
- StateGraph
- Electrical
- Magnetic
- Mechanics
- Fluid
- Media
- Thermal
- Math
- ComplexMath
- Utilities
- Constants
- Icons
- SIunits

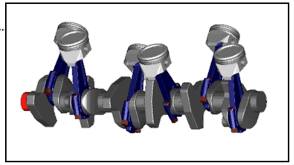
Example: **detailed vehicle model**

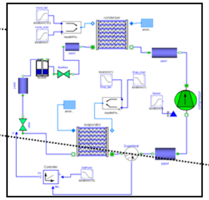
- **Vehicle dynamics** (3-dim. mechanics)
- **Drive trains** (1-dim. mechanics)
- **Hydraulics**
- **Combustion**
- **Air Conditioning** (Thermofluid systems)
- **Electrical/electronic systems**
- **Electrical machines**
- **Hierarchical state machines**
- **Control** (Input/output blocks, ...)

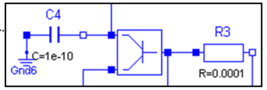

courtesy: Modelon AB

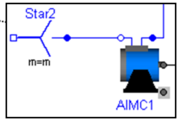


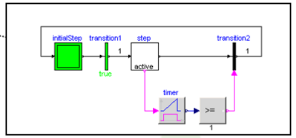





courtesy Modelon AB



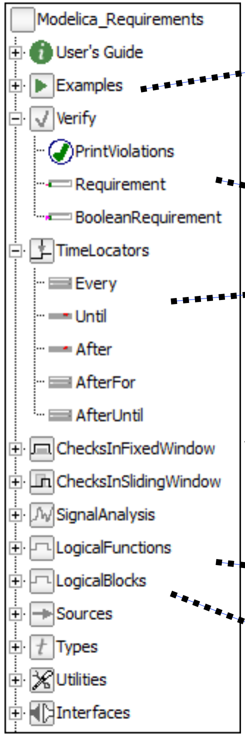




26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Modelica_Requirements library

(Developed in MODRIO Project)



Modelica_Requirements
- User's Guide
- Examples
- Verify
  - PrintViolations
  - Requirement
  - BooleanRequirement
- TimeLocators
  - Every
  - Until
  - After
  - AfterFor
  - AfterUntil
- ChecksInFixedWindow
- ChecksInSlidingWindow
- SignalAnalysis
- LogicalFunctions
- LogicalBlocks
- Sources
- Types
- Utilities
- Interfaces

Many examples

Defining requirements

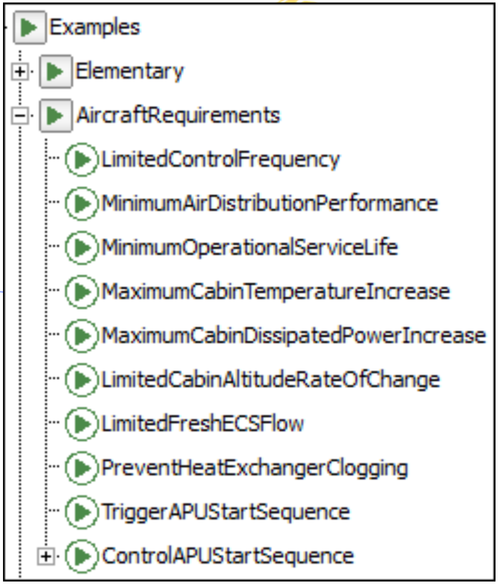
Define when to check a property

Check property in a **fixed time window**

Check property in a **sliding time window**

2/3-valued logic operators **without memory**

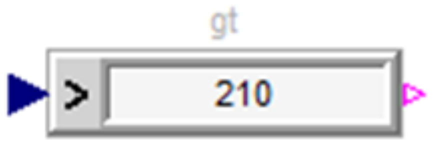
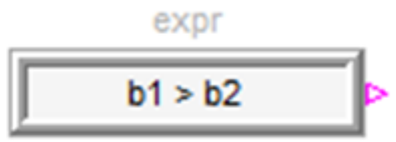
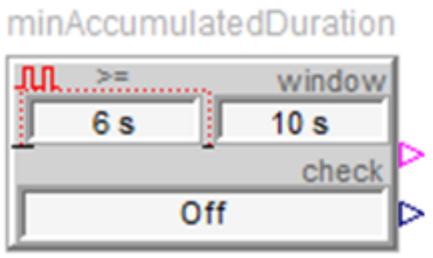
2/3-valued logic operators **with memory**



Examples
- Elementary
- AircraftRequirements
  - LimitedControlFrequency
  - MinimumAirDistributionPerformance
  - MinimumOperationalServiceLife
  - MaximumCabinTemperatureIncrease
  - MaximumCabinDissipatedPowerIncrease
  - LimitedCabinAltitudeRateOfChange
  - LimitedFreshECSFlow
  - PreventHeatExchangerClogging
  - TriggerAPUStartSequence
  - ControlAPUStartSequence

Adapted from: Otter et al: Formal Requirements Modeling for Simulation-Based Verification, Modelica’2015

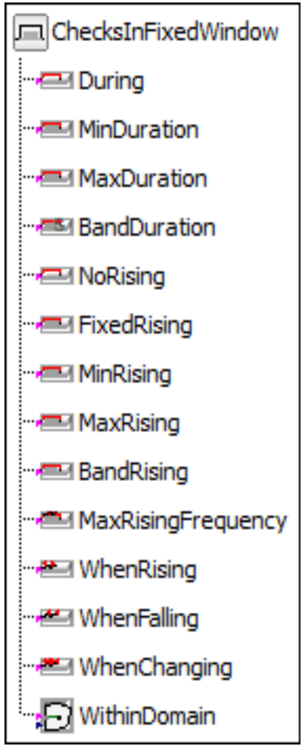
Graphical layout

$y = u > 210$	
$y = b1 > b2$	
y = true when off has been true for more than 6 accumulated seconds during any 10 second time window.	 minAccumulatedDuration >= window 6 s 10 s check Off

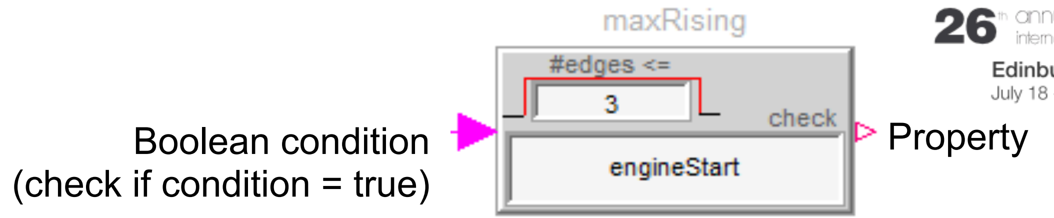
Parameters of components are shown in its icon

Adapted from: Otter et al: Formal Requirements Modeling for Simulation-Based Verification, Modelica'2015

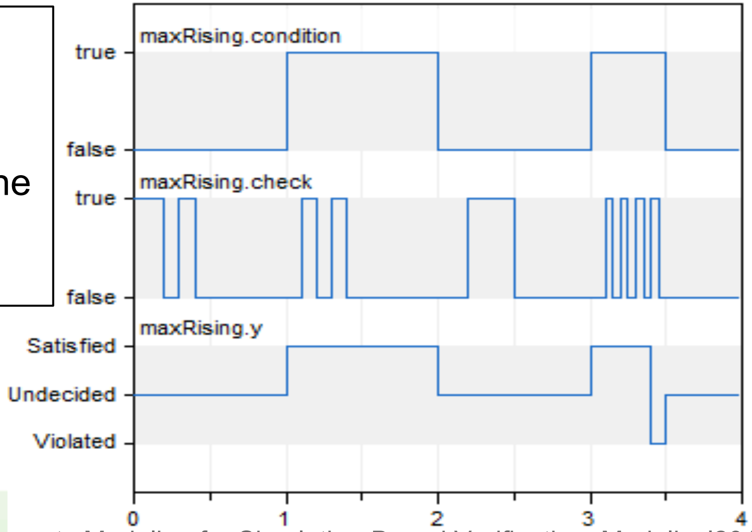
Checks in fixed time windows


ChecksInFixedWindow
- During
- MinDuration
- MaxDuration
- BandDuration
- NoRising
- FixedRising
- MinRising
- MaxRising
- BandRising
- MaxRisingFrequency
- WhenRising
- WhenFalling
- WhenChanging
- WithinDomain

maxRising
In every duration where the Boolean input condition is true, the number of rising edges of the Boolean input check is not allowed to exceed its limit.



Boolean variable that is checked



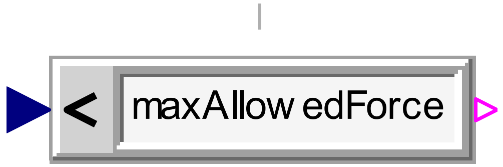
Adapted from: Otter et al: Formal Requirements Modeling for Simulation-Based Verification, Modelica'2015

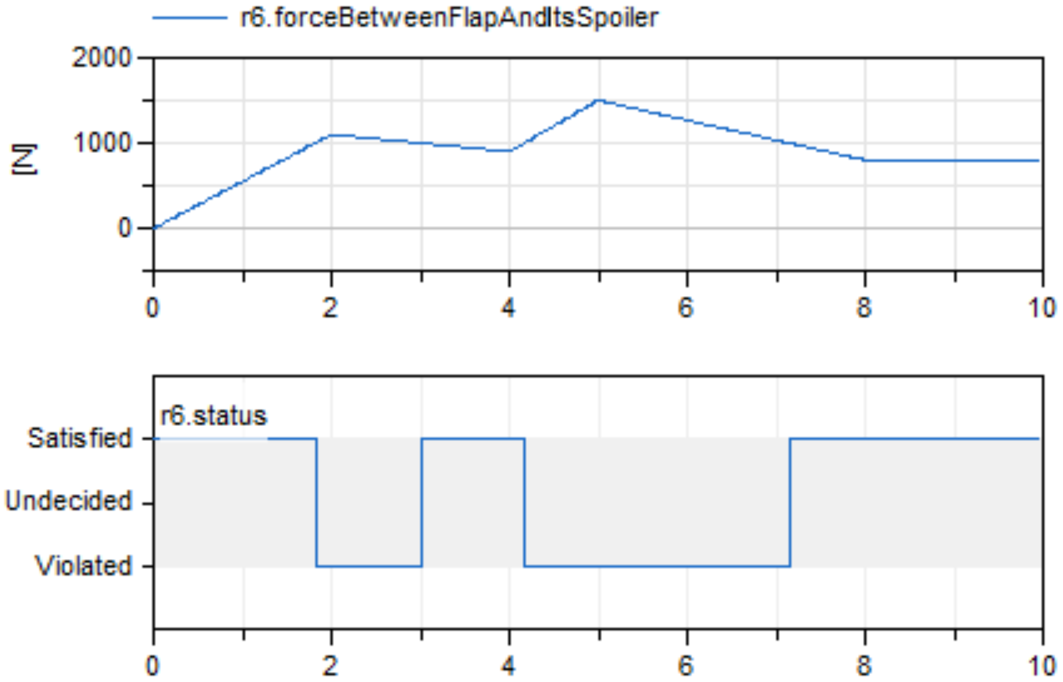
Example

- “Force between a flap and its spoiler shall be less than 1000N”.

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Input: Force between flap and spoiler (Real)

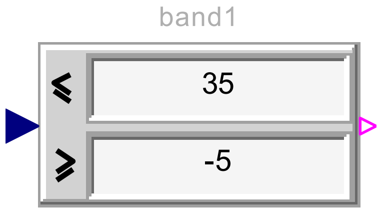


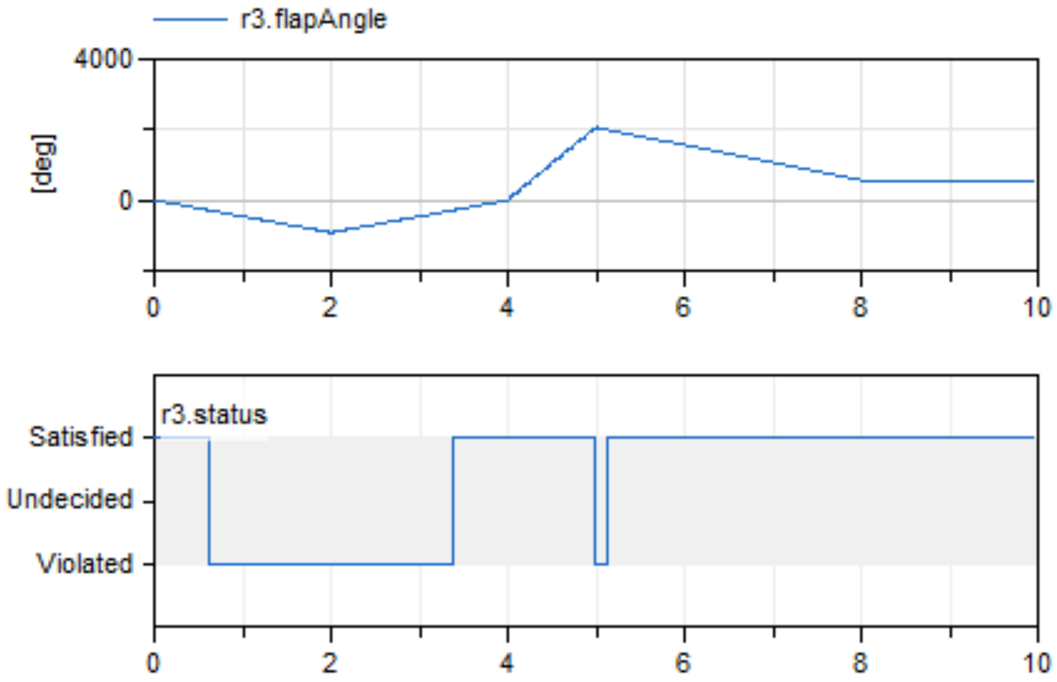


Example

- "The flap angle shall not exceed the range [-5°;35°]".

<span style="color:red">Input</span>: Flaps angle (Real)

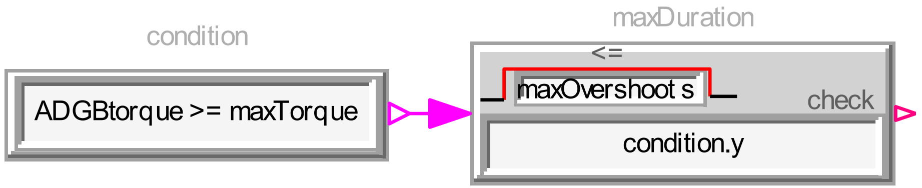


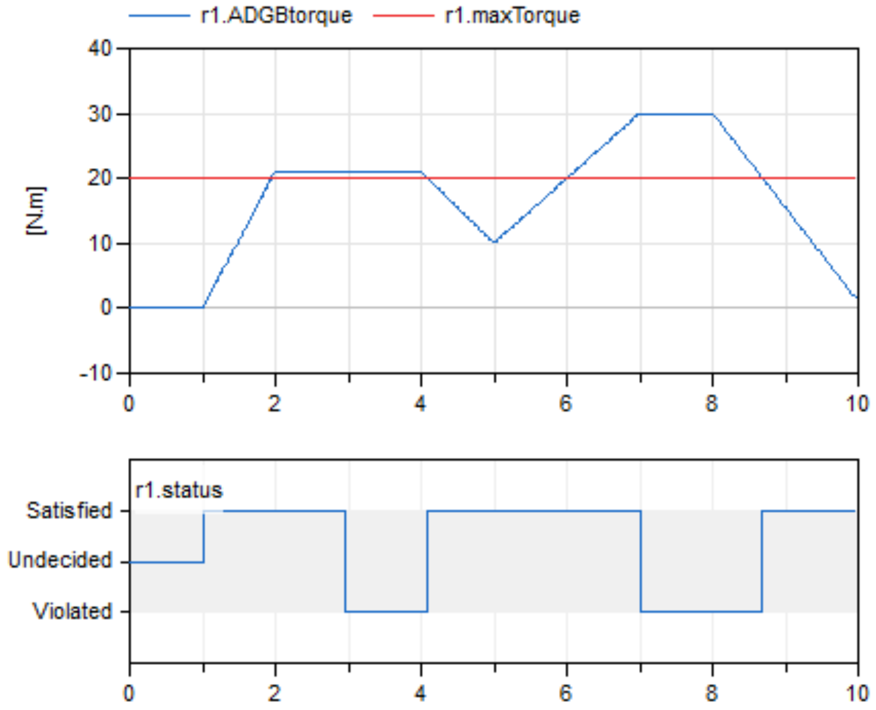


Example

- “The torque of any electrical motor shall not be greater than 20 Nm for more than 1 sec.”

Input: Torque of electrical motors (Real)

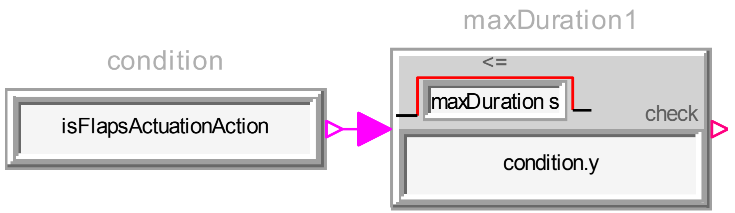


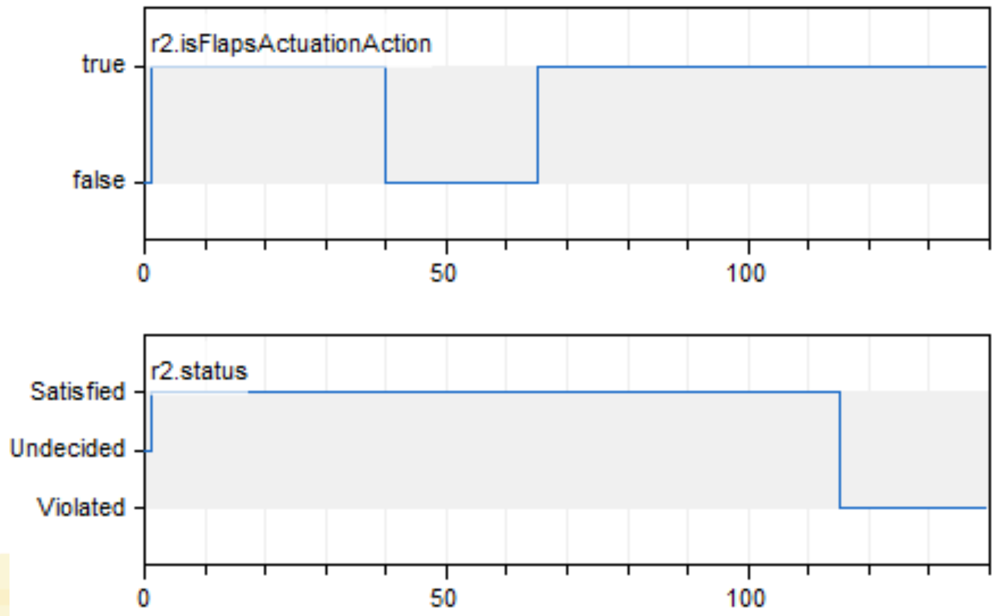


Example

- "The time of any action of flaps actuation (extension/ retraction) shall be less than 50 sec."

Input: Flaps action is happening (Boolean)





System design and scenario models

- **Formalize Designs** - Characteristics of **system design** models:
  - Provide data needed for requirement violation monitors
  - The model shall represent adequately the system (balance between simlification and computation costs)
  - In addition to normal behavior, include degraded or failure behavior, and include models of the system environment

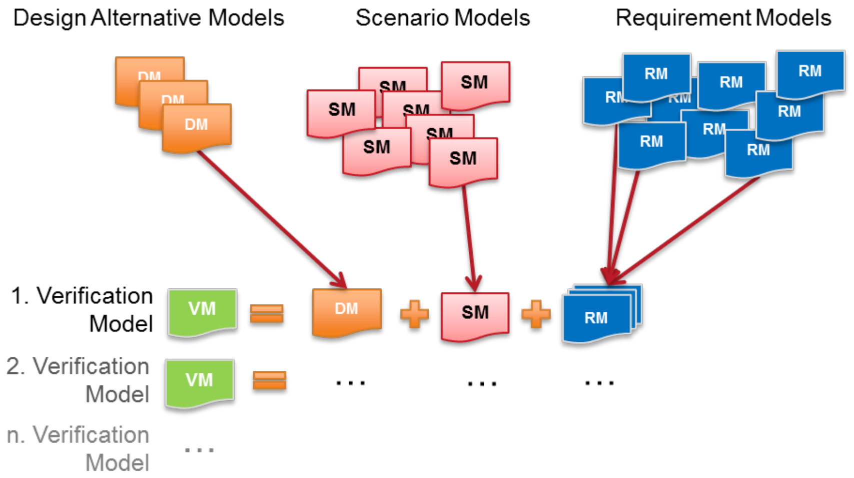
- **Formalize Scenarios -** Characteristics of **scenario** models:
  - Scenarios shall test the system and enable testing of multiple requirements
  - Only contain the course of actions to stimulate the system model, no need for a verdict (verdicts are in requirement violation monitors)
  - Scenario models shall be as independent as possible of different design alternatives (reusability of scenarios)

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

MODEL COMPOSITION

What can we automate?

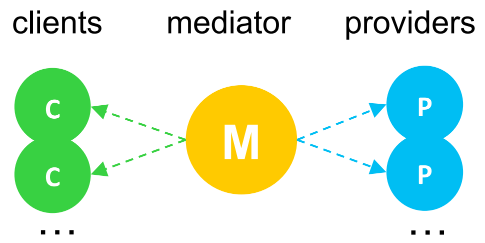
- ☹ Translating natural-languages requirements into violation monitors requires correct interpretation and understanding of the requirement
- ☹ Modeling of system designs and scenarios requires creative human engineering capabilities
- ☺ Creation of verification models, simulation and results interpretation



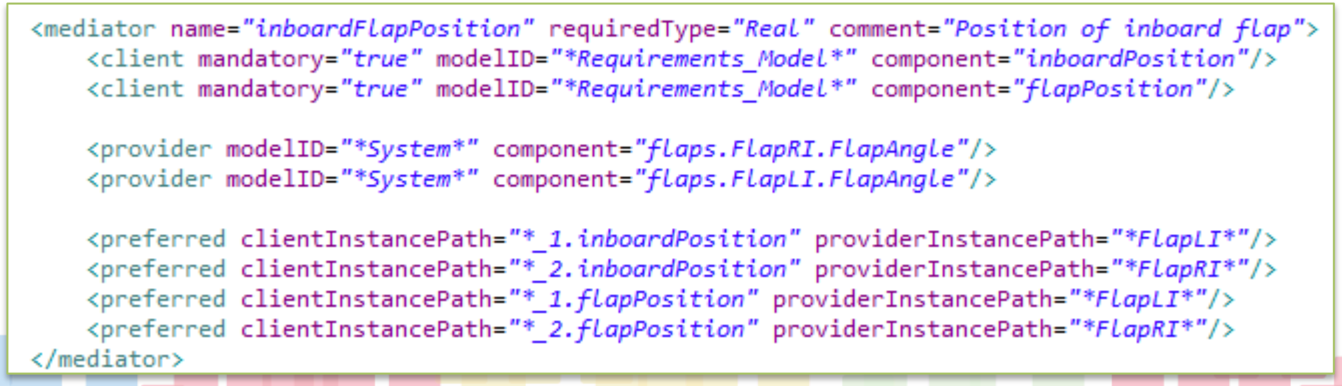
– **How to find such combinations and generate verification models automatically?**

Bindings specification: Basic idea

- Some models require data: *Clients*
- Some models can provide require data: *Providers*
- However, clients and providers do not know each other a priori
- *Mediators* relate a number of clients to a number of providers

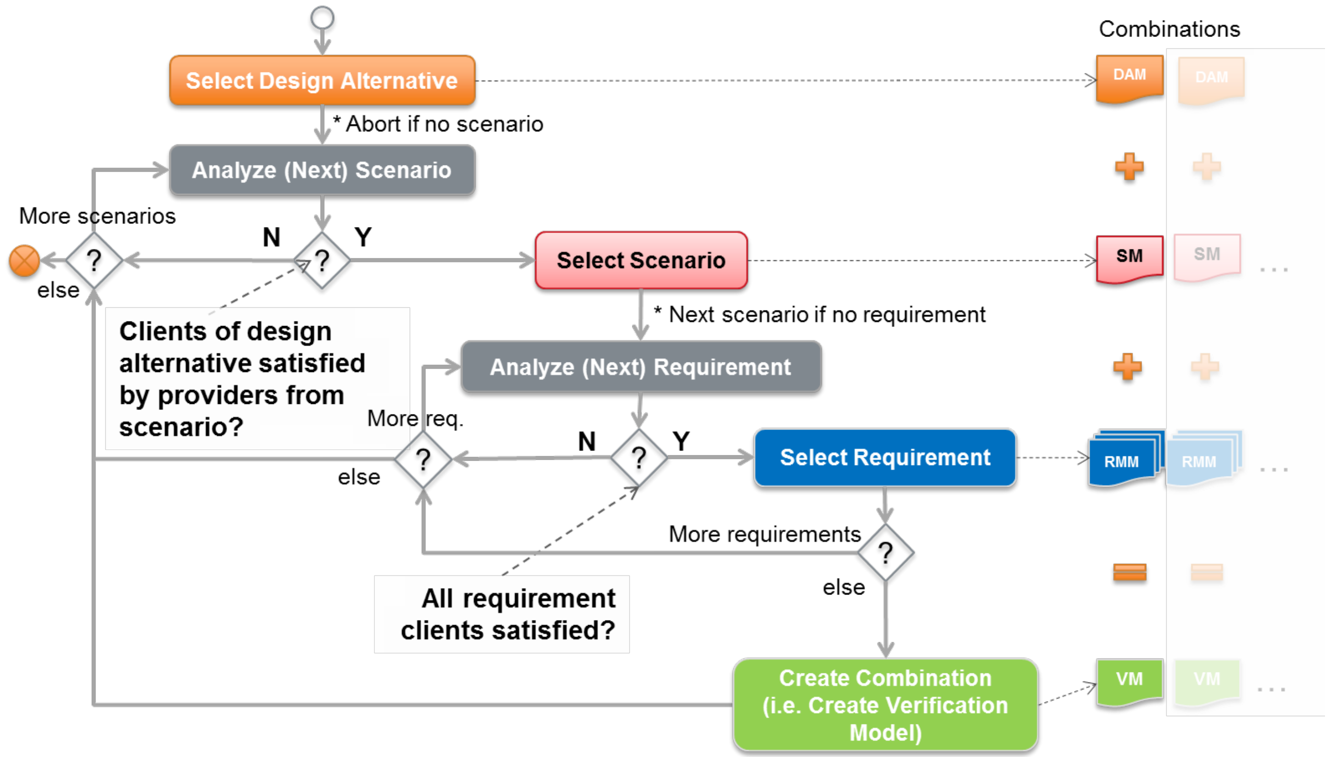


Example:



```xml
<mediator name="inboardFlapPosition" requiredType="Real" comment="Position of inboard flap">
    <client mandatory="true" modelID="*Requirements_Model*" component="inboardPosition"/>
    <client mandatory="true" modelID="*Requirements_Model*" component="flapPosition"/>

    <provider modelID="*System*" component="flaps.FlapRI.FlapAngle"/>
    <provider modelID="*System*" component="flaps.FlapLI.FlapAngle"/>

    <preferred clientInstancePath="*_1.inboardPosition" providerInstancePath="*FlapLI*"/>
    <preferred clientInstancePath="*_2.inboardPosition" providerInstancePath="*FlapRI*"/>
    <preferred clientInstancePath="*_1.flapPosition" providerInstancePath="*FlapLI*"/>
    <preferred clientInstancePath="*_2.flapPosition" providerInstancePath="*FlapRI*"/>
</mediator>
```

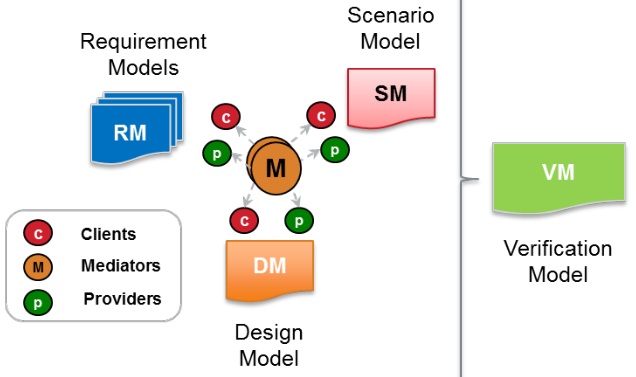
Algorithm for model composition


Combinations
Select Design Alternative
DAM
DAM
* Abort if no scenario
Analyze (Next) Scenario
More scenarios
N
Y
?
?
else
Select Scenario
SM
SM
Clients of design alternative satisfied by providers from scenario?
* Next scenario if no requirement
Analyze (Next) Requirement
More req.
N
Y
?
?
else
Select Requirement
RMM
RMM
More requirements
?
else
All requirement clients satisfied?
Create Combination (i.e. Create Verification Model)
VM
VM
. . .

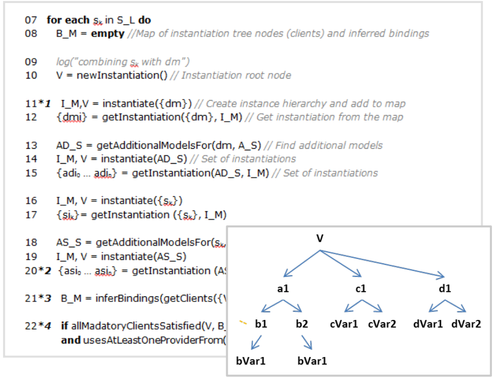
26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Framework for automation

Potential model interactions:
Bindings specification

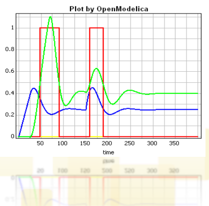


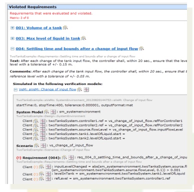
Algorithm for model composition



```
07 for each sₖ in S_L do
08    B_M = empty //Map of instantiation tree nodes (clients) and inferred bindings

09    log("combining sₖ with dm")
10    V = newInstantiation() // Instantiation root node

11*1  I_M,V = instantiate({dm}) // Create instance hierarchy and add to map
12    {dmi} = getInstantiation({dm}, I_M) // Get instantiation from the map

13    AD_S = getAdditionalModelsFor(dm, A_S) // Find additional models
14    I_M, V = instantiate(AD_S) // Set of instantiations
15    {adi₀ … adiₙ} = getInstantiation(AD_S, I_M) // Set of instantiations

16    I_M, V = instantiate({sₖ})
17    {siₖ}= getInstantiation ({sₖ}, I_M)

18    AS_S = getAdditionalModelsFor(sₖ,
19    I_M, V = instantiate(AS_S)
20*2  {asi₀ … asiₙ} = getInstantiation (A

21*3  B_M = inferBindings(getClients({V

22*4  if allMadatoryClientsSatisfied(V, B_
      and usesAtLeastOneProviderFrom(
```

Rules for automated conclusion on simulation results; report generation





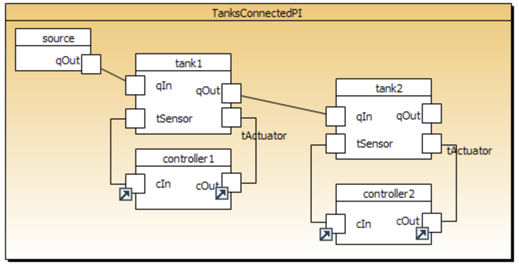
Role		Task
Requirement Analyst		Formalize Requirements
System Designer		Formalize Designs
Tester		Formalize Scenarios
	AUTOMATED	Create Verification Models
	AUTOMATED	Execute and Create Report
System Designer, all		Analyze Results

Former case studies

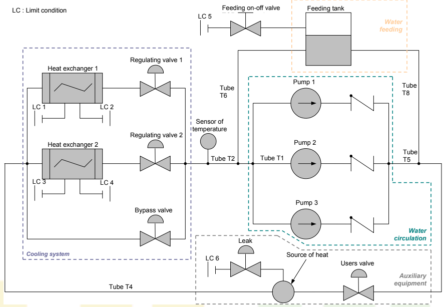
26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

- ModelicaML prototype (see www.openmodelica.org) was used to illustrate the applicability of the new method to examples from industry (3 public case studies)
- Case studies start with sets of natural-language requirements and show how they are translated into violation monitor models
- Then, designs and verification scenarios are modeled in ModelicaML, and simulation models are composed and simulated automatically
- Simulation results are used for automatically drawing conclusions on requirement violations

Two-Tank System



Power Plant Cooling System

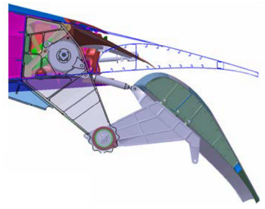


Fuel Display System

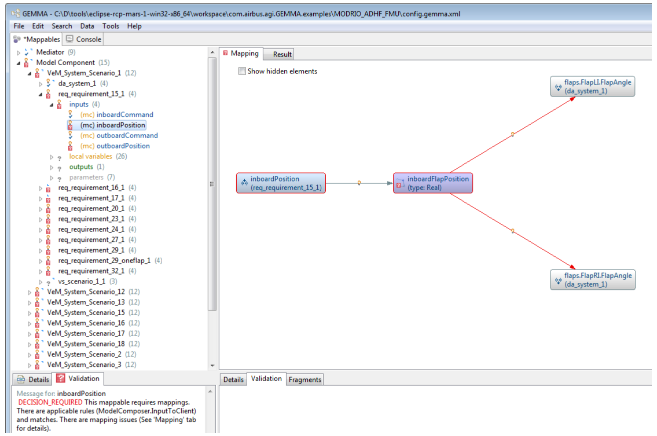
Composing models using GEMMA

- Models modelled in different tools and exported as FMUs (Functional Mock-Up Units)
- GEMMA (generic modular mapping framework) was used to:
  - Combine and connect FMUs based on rules such as equality, similarity or compatibility of input/output name, type and direction, and using the bindings specification
  - Determine how often a particular requirement should be instantiated
  - User is involved when automatic resolving of model connections is not possible

- Output: Verification models (i.e., executable Modelica models) for batch simulation for compiling verification report



GEMMA
Generic
Modular
Mapping
Framework
AIRBUS GROUP INNOVATIONS



GEMMA - C:\D\tools\eclipse-rcp-mars-1-win32-x86_64\workspace\com.airbus.agi.GEMMA.examples\MODRIO_ADHF_FMU\config.gemma.xml

```
Mediator (9)
Model Component (15)
  VeM_System_Scenario_1 (12)
    da_system_1 (4)
    req_requirement_15_1 (4)
      inputs (4)
        (mc) inboardCommand
        (mc) inboardPosition
        (mc) outboardCommand
        (mc) outboardPosition
      local variables (26)
      outputs (1)
      parameters (7)
    req_requirement_16_1 (4)
    req_requirement_17_1 (4)
    req_requirement_20_1 (4)
    req_requirement_23_1 (4)
    req_requirement_24_1 (4)
    req_requirement_27_1 (4)
    req_requirement_29_1 (4)
    req_requirement_29_oneflap_1 (4)
    req_requirement_32_1 (4)
    vs_scenario_1_1 (3)
  VeM_System_Scenario_12 (12)
  VeM_System_Scenario_13 (12)
  VeM_System_Scenario_15 (12)
  VeM_System_Scenario_16 (12)
  VeM_System_Scenario_17 (12)
  VeM_System_Scenario_18 (12)
  VeM_System_Scenario_2 (12)
  VeM_System_Scenario_3 (12)
```

Message for: inboardPosition
DECISION_REQUIRED This mappable requires mappings. There are applicable rules (ModelComposer.InputToClient) and matches. There are mapping issues (See 'Mapping' tab for details).

File
Edit
Search
Data
Tools
Help
*Mappables
Console
Mapping
Result
Show hidden elements
inboardPosition (req_requirement_15_1)
inboardFlapPosition (type: Real)
flaps.FlapLI.FlapAngle (da_system_1)
flaps.FlapRI.FlapAngle (da_system_1)
Details
Validation
Fragments

Conclusion

- vVDR is applicable to realistic problems from aerospace domain
  - New models be developed independently
  - Contributes to model reuse: No need for defining explicit model-interfaces or modifying existing models
  - Supports uncovering incompleteness or inconsistencies specifications and contributes to improving requirements specifications and designs

- Case Study:
  - Using the Modelica_Requirements library reduces modelling effort when formalizing natural language requirements into executable violation monitor models
  - The presented approach for automated model composition significantly decreases effort for low added-value and error-prone tasks (e.g., manually connecting models (few hundred connections or parameters), or interpreting simulation results)

Thank you for your attention!

26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Airbus Group Innovations
Wladimir Schamai
Wladimir.Schamai@airbus.com