Industrial Use Cases for Requirements Verification and Model Composition in ModelicaML

OPENPROD-Project Case Study Results

Wladimir SCHAMAI  (EADS Innovation Works, Germany)
Audrey JARDIN  (EDF - R&D, France, OPENPROD Partner)
Daniel BOUSKELA  (EDF - R&D, France, OPENPROD Partner)

MODPROD 2013  
Feb. 2013
System Description

- The **SRI system** (Intermediate Cooling System) is located in the turbine hall of a nuclear power plant. Its main purpose is to ensure the cooling of some auxiliary equipment which belongs to the conventional part of the plant and requires demineralized water (e.g. alternator, circulation pump…)

- Three main functions:
  - **Cooling**, ensured by two plate heat exchangers and their control valves, and one bypass valve;
  - **Water circulation**, ensured by three pumps in parallel, the third one being used as backup in case of the failure of one of the other two pumps, and hence stopped under normal operation conditions;
  - **Water feeding**, ensured by one tank and its water supply.

- The **I&C** of the SRI system functions:
  - Control of the temperature set point at the outlet of the heat exchangers;
  - Control of the tank water level between two limits;
  - Logics of the startup of the backup pump;
System Description

LC : Limit condition

Heat exchanger 1
Heat exchanger 2
Regulating valve 1
Regulating valve 2
Bypass valve

Pump 1
Pump 2
Pump 3
Sensor of temperature

Feeding on-off valve
Feeding tank

Tube T1
Tube T2
Tube T6
Tube T8
Tube T5

Tube T4

Source of heat
Users valve
Leak

LC 1
LC 2
LC 3
LC 4
LC 5
LC 6

Cooling system
Water feeding
Water circulation
Auxiliary equipment

TOR alimentation
Feeding on-off valve

LC 1
LC 2
LC 3
LC 4
LC 5
LC 6

Water feeding
Water circulation
Auxiliary equipment

LC : Limit condition
System Requirements

- **#002**: The set point of the SRI water temperature must be held at a minimum value of 17° C.

- **#003**: In a normal operating mode, the water temperature of the SRI circuit should be between $T_s - e$ and $T_s + e$ ($T_s$ : set point temperature).

- **#0083**: A pump must not start more than 3 times per hour.

- **#013**: In a normal operating mode, there must not be less than 2 operating pumps during more than 2s.

- **#007**: The water temperature must not vary more than 10° C/hour.
Case Study Purpose

• Formalize system requirements in order to enable detecting their violations:
  • when simulating a system (e.g. for design verification)
  • or running the real system (e.g. for operation monitoring)
  • offline (e.g. post-processing of recorded data)
  • and online (in real-time)

• Show how to do it using the vVDR method and the ModelicaML environment
### vVDR Case Study

**Goal:** Enable design verification using models

#### vVDR Method Overview

<table>
<thead>
<tr>
<th>Actor</th>
<th>Task</th>
<th>Created Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Analyst</td>
<td>Formalize Requirements</td>
<td>Requirement Monitor Models</td>
</tr>
<tr>
<td>System Designer</td>
<td>Formalize Designs</td>
<td>Designs Alternative Models</td>
</tr>
<tr>
<td>System Tester</td>
<td>Formalize Scenarios</td>
<td>Scenario Models</td>
</tr>
<tr>
<td>AUTOMATED</td>
<td>Create Verification Models</td>
<td>Verification Models</td>
</tr>
<tr>
<td>AUTOMATED</td>
<td>Execute and Create Report</td>
<td></td>
</tr>
<tr>
<td>System Designer</td>
<td>Analyze Results</td>
<td></td>
</tr>
</tbody>
</table>

**Key Terms:**
- **RMM**: Requirement Monitor Models
- **DAM**: Designs Alternative Models
- **SM**: Scenario Models
- **VM**: Verification Models

---

**Note:**
- The diagram illustrates the steps involved in formalizing requirements, designs, and scenarios, followed by the creation of verification models and the execution of verification processes. The goal is to enable design verification using models, with roles assigned to different actors for each step.

---

**References:**
- EADS Innovation Works
- Technical documentation on vVDR methodology
ModelicaML

- ModelicaML integrates a subset of UML and Modelica to leverage standardized graphical notations of UML for system modeling and the simulation power of Modelica.

- ModelicaML tool supports vVDR.
- It enables engineers to formalize:
  - Requirements
  - Design alternatives
  - Scenarios
  - Bindings (for automated model composition)
REQUIREMENTS

FORMALIZATION
Method for Formalizing Requirements

Formalization purpose:
- For validation: Detect ambiguity and incompleteness in textual requirements, check coherence between the different requirements
- For verification: Detect violation of requirement (focus of this case study)

Formalization method:
1. Read requirement statement
2. Identify quantifiable properties mentioned in the requirement statement
3. Identify pre-conditions (if any) when the requirement should be evaluated
4. Express when the requirement is violated and when not
   - I.e., express when to set the “status” attribute to:
     - 0 = not evaluated (as long as pre-conditions (if any) are not met)
     - 1 = not violated (implies that the requirement was evaluated)
     - 2 = violated (implies that the requirement was evaluated)
Formalizing Requirements

- #002: The set point of the SRI water temperature must be held at a minimum value of 17°C.

Quantifiable properties:

- input Temp_C setPoint
- parameter Temp_C setPointMinimum = 17
- output Integer status = if setPoint < setPointMinimum

The set point of the SRI water temperature … … value of 17°C
**Formalizing Requirements**

- **#002**: The set point of the SRI water temperature must be held at a minimum value of 17°C.

No pre-condition, evaluated at any time, no need to set status to 0 (=not evaluated)

\[
\text{status: } 0 = \text{not evaluated}; 1 = \text{not violated}; 2 = \text{violated};
\]
**Formalizing Requirements**

- **#0083**: A pump must not start more than 3 times per hour.
  - No pre-condition
  - This requirement will need to be evaluated for each pump in the system

In order to monitor this requirement we need to access values of the past 1h.

In Modelica we could use the `delay()` operator. However, it would need to store values for the last 1h. Moreover, it would detect the violation 1h too late!
Formalizing Requirements

- #0083: A pump must **not start more than** 3 times per hour.

Approach implemented in this case study:
- Store the last 4 activations (timestamps) of a pump
- Compute the time difference between the 1\(^{st}\) and the 4\(^{th}\) activation
- If the difference is less than 1h -> violation

Violation detected correctly

Pump was turned on

1 hour  1 hour
#0083: A pump must not start more than 3 times per hour.

An array of size 4 that is filled with timestamps of last pump activations. The array is updated each time a pump is activated:
- All entries are shifted by one position up (the first gets deleted)
- The new entry is inserted at the last position

Quantifiable properties:

- “A pump must ... start”
- “… more than 3 times …”
- “… per hour. (i.e. 3600sec.)”
Formalizing Requirements

- #0083: A pump must not start more than 3 times per hour.
Formalizing Requirements

- #0083: A pump must not start more than 3 times per hour.

Now we are able to compute the status
• #0083: A pump must **not start more than 3 times per hour.**
#007: The **water temperature** must not vary more than **10° C/hour**.

- **Approach:** Determine the difference between the lowest and the highest temperature value within 1 h sliding window.
- If the difference is more than **10° C** -> violation

```
• Rq 007 - Varying of water temperature
  • input Temp_C temperature
  • parameter Real allowedDifference = 10
  • parameter Duration observationPeriod = 3600
  • output Integer status = if abs(min(sampledPoints) - max(sampledPoints)) > .
  • Temp_C[... sampledPoints
  • parameter Integer numberOfSamples = 360
  • parameter Duration samplePeriod = observationPeriod / numberOfSamples
  • Real minTempWithinObservationPeriod
  • Real maxTempWithinObservationPeriod
  • eq: Initialize
  • alg: Update samples
  • updateSlidingWindow
```

Quantifiable properties:
- “...water temperature ...”
- “... 10° C”
- “…per hour.”
Formalizing Requirements

- #007: The water temperature must not vary more than 10°C/hour.

- We cannot rely on continuous temperature function (e.g. because of sampled sensor value)

- The temperature needs to be sampled: An array of temperature values for the past 1h
  - How to compromise on the sample rate?
    - Let domain expert decide for what is the minimum acceptable sampling rate by considering how fast the temperature may change.
Formalizing Requirements

- #007: The water temperature must not vary more than $10^\circ$ C/hour.

Sampled temperature values

```
when sample(0, samplePeriod) then
  /* update the samples array */
  sampledPoints := updateSlidingWindow(
    numberOfSamples,
    sampledPoints,
    temperature);

  minTempWithinObservationPeriod := min(sampledPoints);
  maxTempWithinObservationPeriod := max(sampledPoints);
end when:
```
• #007: The water temperature must not vary more than 10°C/hour.
Testing Formalized Requirements

- #007: The **water temperature** must not vary more than 10° C/hour.

![Graph showing water temperature variation with status indicators.](image)

**Status:** 0 = not evaluated; 1 = not violated; 2 = violated;

Dymola demo version, see www.Dymola.com
#007: The water temperature must not vary more than 10° C/hour.
DESIGN MODEL
Design model was built using ThermoSysPro library (developed by EDF) which is open source and freely available on request.
**Scenari**

- **Sc01 Normal operating mode:** No failure, two pumps on, two heat exchangers on operation
- **Sc02 Incorrect setting of the temperature regulation**
- **Sc03 Pump switching:** Pumps start several times due to occurrences of sequential failures
- **Sc04 Pump recovery:** If a pump stops because of a failure, the first available pump starts; if it is not on after 2 seconds, the next available pump is started
- **Sc05 Cold winter:** External weather is too cold so that the SRI keeps a water temperature above 15°C.
- **Sc06 Overheating:** The auxiliary equipment delivers a too large amount of heat for which SRI isn't sized.
- **Sc07 Fast water filling:** In case of a dysfunction in the automatic feed water system, the manual valve should allow the water filling of the circuit in less than 2 hours
Scenarios (created in ModelicaML)

- **Sc01 Normal operating mode**
  - parameter Real SEN_Temperature = 285
  - output Boolean Feedwater_Valve_Failure.y.signal = false
  - output Boolean Command_Valve_Feedwater.y.signal = false
  - output Boolean Pump1Failure.y.signal = false
  - output Boolean Pump2Failure.y.signal = false
  - output Boolean Pump3Failure.y.signal = false
  - output Real SetPoint_Temperature.y.signal = 290.15
  - output Boolean Exchanger1Valve.y.signal = true
  - output Boolean Exchanger2Valve.y.signal = true
  - output Real Heat_AuxiliaryEquipment.y.signal = 25e6
  - output Real Leak.y.signal = 1
  - output Real ManualValve_Circulation.y.signal = 0.36
  - comments (1)
    - <<Annotation>> <Comment> No failure, two pumps on, two heat exchangers on operation

- **Sc02 Incorrect setting of the temperature regulation**

- **Sc03 Pump switching**

- **Sc04 Pump recovery**

- **Sc05 Cold winter**
  - parameter Real Real SEN_Temperature = 273.15
  - output Real Heat_AuxiliaryEquipment.y.signal = 0
  - comments (1)
    - <<Annotation>> <Comment> External weather is too cold so that the SRI keeps a water temperature above 15°C.

- **Sc06 Overheating**

- **Sc07 Fast water filling**
AUTOMATED MODEL COMPOSITION
**Bindings Concept**

- **Binding** enables the automation of verification model composition

- Value Bindings include the definition of:
  - **Client** (component that requires data from other components)
  - **Provider** (component that provides data for other components)
  - **Mediator** (mediates between clients and providers)

- Depending on which mediators and providers are in place we can:
  - Determine which clients can be satisfied
  - **Find valid combinations** and generate verification models
  - Generate **binding** code for client components in verification models
Bindins for Requirements

- Client
  - temperature
    - mand. client, Temp_C, ModelicaMLModel::Requirements::Varying of water temperature
  - waterTemperatureOfSRICircuit
    - mand. client, Temp_C, ModelicaMLModel::Requirements::Water temperature in normal mode

- Provider from design model
  - T
  - provider, AbsoluteTemperature, ThermoSysPro::WaterSteam::Sensors::SensorT

- Monitor clients
  - is in normal operating mode
  - pump is on
  - status of all pumps
  - pump is on

- Provider from design model
  - marche
  - provider, InputLogical, ThermoSysPro::ElectroMechanics::Machines::SynchronousMotor
  - set point of the SRI water temperature
  - pumpsStatus
  - pumpsStatus
  - pumpsStatus
  - pumpsStatus
  - pump is on
  - status of all pumps
  - pump is on
  - status of all pumps
Bindedings for Design and Scenarios

- Command_Valve_Feedwater
  - mediator, Boolean, clients (1), providers (2)
  - Command_Valve_FeedWater - client, BooleanInput, Design::SRIn4_v3::Environment
  - Command_Valve_FeedWater.y.signal - provider, Boolean, ModelicaMLModel::Scenarios::Sc01 Normal operating mode
  - Command_Valve_FeedWater.y.signal - provider, Boolean, ModelicaMLModel::Scenarios::Sc07 Fast water filling

- Exchange2Valve
  - mediator, Boolean, clients (1), providers (1)

- Exchanger1Valve
  - mediator, Boolean, clients (1), providers (1)

- Feedwater_Valve_Failure
  - mediator, Boolean, clients (1), providers (2)

- Heat_AuxiliaryEquipment
  - mediator, Real, clients (1), providers (4)

- Leak
  - mediator, Real, clients (1), providers (2)

- ManualValve_Circulation
  - mediator, Real, clients (1), providers (1)

- Pump1Failure
  - mediator, Boolean, clients (1), providers (1)

- Pump2Failure
  - mediator, Boolean, clients (1), providers (2)

- Pump3Failure
  - mediator, Boolean, clients (1), providers (1)

- Set point of the SRI water temperature
  - mediator, Real, clients (1), providers (2)

- SetPoint_Temperature
  - client, RealInput, Design::SRIn4_v3::Environment

- SetPoint_Temperature.y.signal
  - provider, Real, ModelicaMLModel::Scenarios::Sc01 Normal operating mode

- SetPoint_Temperature.y.signal
  - provider, Real, ModelicaMLModel::Scenarios::Sc02 Incorrect setting of the temperature regulation
Automated Model Composition

1. Select a design model
2. Find scenarios that can stimulate this design model and requirements that are implemented
3. Create combinations as models and bind all components
Generated Verification Models

- Instantiated 'aVeM: Sc01 Normal operating mode' (4867 nodes)
  - add_0_NormalModeCalculation (3)
  - req_002_minimum_set_point_of_the_sri_water_temperature (3)
  - req_003_water_temperature_in_normal_mode (5)
  - req_007_varying_of_water_temperature (9)
  - req_0083_0_number_of_times_a_pump_is_switched_on_per_hour (5)
    - Time activationTimePoints[...]
    - Integer maxNumberOfSwitchOnToCount = 3
    - (mand. client), input pumpIsOn = sm_global_encapsulated.sRI_v3_env.Moteur1.marche.signal
    - Duration slidingWindowDuration = 3600
    - output status
  - req_0083_1_number_of_times_a_pump_is_switched_on_per_hour (5)
  - req_0083_2_number_of_times_a_pump_is_switched_on_per_hour (5)
  - req_013_number_of_operating_pumps (8)
    - (mand. client), input isNormalOperatingMode = add_0_NormalModeCalculation.isInNormalOperation
    - Duration maxDuration = 2
    - Integer minNumberOfActivePumps = 2
    - (mand. client), input numberOfPumps = 3
    - Integer numberOfPumpsCurrentlyActive
      - (mand. client), input pumpsStatus[...] = {sm_global_encapsulated.sRI_v3_env.Moteur1.marche.signal, sm_global_encapsulated.sRI_v3_env.Moteur2.mar
      - output status = if not isNormalOperatingMode then 0 else if isNormalOperatingMode and tooFewPumpsDetectedAt > -1 and (time-tooFewPumpsDe
      - Time tooFewPumpsDetectedAt
  - sm_global_encapsulated (2)
  - vs_sc01_normal_operating_mode (12)

- Additional model
- Requirement monitor
- Generated bindings
- Design
- Scenario
VERIFICATION RESULTS
Verification Session Report
(post-processed simulation result files)

Verification Session Report
Creation date: 2013.01.24 11:57:06

Violated Requirements (2 of 7)
Not Violated Requirements (5 of 7)
Not Evaluated Requirements (0 of 7)
Not Implemented Requirements (2 of 7)
Used Scenarios (7 of 7)
Not Used Scenarios (0 of 7)
CONCLUSION
Conclusion and Lessons Learnt

- The case study proved the applicability of the vVDR method to realistic industrial applications.

- ModelicaML is a promising implementation of the vVDR method but still in a prototyping phase, to be improved regarding usability and stability in order to be used in industrial projects.

- OpenModelica enhancements are needed to enable the simulation of ThermoSysPro library based models.

- Lessons learnt:
  - Formalized requirement should be tested without the system model in order to ensure correctness.
  - Model validity asserts must be included (e.g. assert statements in Modelica, e.g., the requirement monitors are only valid for simulation time > 0).
  - Parameterized requirement monitors can be re-used as templates or library components.

- This work will be continued in the ITEA2 MODRIO project.
  - Stochastic aspects (model uncertainties, tolerances in requirements, ...) should be taken into account.
Thank you for your attention!

Wladimir Schamai
EADS Innovation Works (Germany)
wladimir.schamai@eads.net