Formal Requirements Engineering for Smart Industries

Alexandre Le Borgne, Nicolas Belloir, Jean-Michel Bruel, Thuy Nguyen

* funded by a NeoCampus/IRIT grant
Content

I. **Context**

II. Graphical language Definition

III. Results

IV. Conclusions and future works
Context

• Main Concerns
  • Socio-CPS
  • Numerous factors which may influence the behaviour of the system
  • Power plant => 60-years life-cycle
  • Intermittent renewable power production
  • Numerous and wide uncertainties
  • Completeness of requirements
  • Ensure stability of the French power grid
  • Early optimization
  • FORM-L (FOrmal Requirement Modeling Language)
Limitations

Functional Requirements and Use Cases, http://www.bredemeyer.com

SysML requirement diagram, http://formalmind.com

KAOS requirements, http://foswiki.cs.uu.nl/
FORM-L

• Define the **envelope** of the system
  • Avoid over constraining
    • The model **must not** describe a solution
  • Model the **environment** of the system as **assumptions**.

• Formal expression of requirements
  • **Simulation**

• Complex language
  • Several ways for defining requirements $\rightarrow r1 \Leftrightarrow r2$

---

Pumps must not cavitate as long as they are operating.

```plaintext
property Model REQ

class Pump
    external Boolean cavitates;
    external Boolean inOperation;
end Pump;

external Pump {} pumps;

requirement r1 =
    forall p in pumps
    during p.inOperation
    check no p.cavitate;

requirement r2 {} = {
    forall pump in pumps
    | during p.inOperation
    check no p.cavitates
};
end REQ;
```
Problem

• No FORM-L editor
• Difficult to write for untrained people

• Need of a graphical language
  • Visualize FORM-L concepts
  • Understand FORM-L models

→ FORM-GL
Content

I. Context

II. Graphical language Definition

III. Results

IV. Conclusions and future works
FORM-GL requirements

• Must be easy to understand
  • Untrained people
  • Different background
  • Different native language

• Must be adaptable to the experience of the user
  • Beginner view
  • Expert view

• Must be a compact language
  • No UML-like diagrams
    • Difficult to understand for untrained people
    • Difficult to read when they get to big

• Must be a block language
  • Stick blocks together to get a compact and understandable model

• User might be able to define their own diagram
\(((a1+a2+a3)*a4) \leq \text{fnct}(b1, b2+b3) \) or c3 and d4
FORM-GL

Assembling blocks together

Different representation levels
Tools

**Xtext**
- Eclipse based tool
- Create grammars
  - Eclipse based editor implementing the grammar
  - Generates EMF metamodel of the grammar

**EMF**
- Reference Eclipse tool for designing metamodels
  - DSLs

**Sirius**
- Eclipse based tool
  - Created and maintained by Obeo
  - Creation of graphical modeling tools based on EMF metamodels
  - Xtext integration extension
Content

I. Context

II. Graphical language Definition

III. Results

IV. Conclusions and future works
model REQ

class Pump
    external Boolean cavitates;
    external Boolean inOperation;
end Pump;

external Pump () pumps;

requirement r1 =
    forAll p in pumps
    during p.inOperation
    check no p.cavitates

requirement r2 =
    forAll pump in pumps
    during pump.inOperation
    check no pump.cavitates

end REQ;
Results

- FORM-L Grammar
  - Derive meta-model from the grammar
    - 195 meta-classes
- FORM-L Editor
  - Syntax coloration
  - Syntax verification

```plaintext
property Model req

class Pump
    external Boolean cavitates;
    external Boolean inOperation;
end Pump;

external Pump {} pumps;

requirement r1=
    forall p in pumps
dauring p.inOperation
    check no p.cavitate;

requirement r2 {} = {
    forall pump in pumps
    | during p.inOperation ensure no p.cavitate
}

end req;
```
Results

• FORM-GL Editor
  • Navigation within diagrams
  • Generation of FORM-L from a graphical model
    • And vice-versa
  • Integration of an Xtext embedded editor into the graphical editor
    • Directly change the Xtext expression of a FORM-L element from the graphical editor
Results

• Issues
  • Define and anchor text fields into a node
    ➢ Floating nodes that contain the text
    ➢ But not satisfying regarding the way we want to develop FORM-GL
  • Define topological constraints
    ➢ Get custom node fully resizable keeping defined constraints
Contents

I. Context
II. Graphical language Definition
III. Results
IV. Conclusions and future works
Conclusions and Future Works

• FORM-GL still under development
  • First prototype

• Sirius provides an environment that links graphical and textual aspects

• Suggestions:
  • Introduce topological constraints into Sirius and improve Sirius’ ability to bind blocks together
  • Allow editable text field anchoring
  • Facilitate graphical expression so we can represent operating domains defined by equations or points

• Future work:
  • Grammar refactoring and optimization
  • Model transformation for Modelica
  • Explore model simulation
  • Explore FORM-GL integration to existing languages like SysML
  • Make users able to define their own graphics
    • relations between text and graphic
Temperature $t$

- $155^\circ C$
- $25^\circ C$
- $160^\circ C$
- $70^\circ C$
- $10^\circ C$

Pressure $p$

- $155$ b
- $30$ b
- $25$ b
- $5$ b

$((10.*C \leq t \leq 70.*C) \text{ and } (5.*b \leq p \leq 30.*b)) \text{ or }$

$((70.*C \leq t \leq 160.*C) \text{ and } (25.*b \leq p \leq 30.*b)) \text{ or }$

$((160.*C \leq t \leq 295.8.*C) \text{ and } (25.*b \leq p \leq 155.*b) \text{ and } \left( f_1(t, p) \geq 0. \right) \text{ and } \left( f_2(t, p) \leq 0. \right) \text{ and } \left( f_3(t, p) \leq 0. \right) \text{ and } \left( f_4(t, p) \geq 0. \right) )$
The process can be in this zone, but no more than 5 mn per any time window of 3 hours.
Thank you !!!

QUESTIONS
FORM-L

FOrmal Requirements Modeling Language

- Model envelope of the system
- Avoid over-constraining

**FORM-L main concepts:**
- Variables
- Events
- Properties
- Objects
- RichEvents
- Coordination

**FORM-L Models:**
- Property model
- Object model
- Library
- Behavioral model
- Contract model
- Binding model
- Configuration Model
The process should remain in this zone.
When \( v \) is greater than 205 v, the MPS should be declared on.

When \( v \) is lower than 170 v, the MPS should be declared off.

When \( v \) is lower than 195 v, the MPS should not become on.

When \( v \) is greater than 185 v, the MPS should not become off.
Model-Driven Approach

• Current process:
  • Textual FORM-L (no FORM-L editor)
  • Validation with simulation tool
    → Stimulus
    → Mass simulation
  • Verification of scenarios with Modelica

• Target process:
  • Develop a FORM-GL model
    • Generate FORM-L code
    • Generate entries for Modelica
  • Perform simulation on the generated code, animate the model during simulation