The Common Requirement Modelling Language

15th International Modelica Conference, Oct. 9-11, 2023

Daniel Bouskela, Lena Buffoni, Audrey Jardin,
Vince Molnãr, Adrian Pop, Armin Zavada

Agenda

1. Motivations
2. What is CRML?
3. CRML Methodology by Example
4. CRML Toolchain
5. First Industrial Experiments
6. Conclusions & Outlook

CRML a Language for Verifying Realistic Dynamic Requirements

Ambition: Effective Engineering of Large Cyber-Physical Systems (CPS)

Scope: Cyber-Physical Systems (CPS), especially energy systems

Characteristics

- CPS Projects have often strong **social and environmental impacts**
- They are **long lasting** projects involving numerous stakeholders
- They should obey to **multiple even conflicting requirements**
- **Project performance is a key** as large over costs may be induced quickly due to financial charges (discount rate)

Challenges

- How to focus on conceiving systems more sustainable, trusty and resilient?
- How to solve over-constrained problems? How to coordinate stakeholders efficiently?
- How to specify the right need without going into realization details? How to reconcile innovation with what already exists?
- How to propagate changes in assumptions all over the system design cycle?
- How to evaluate design alternatives efficiently?
- How to perform FMECA all along design lifecycle?
- How to justify and document design choices for future generations?

Examples of Challenges Related to Energy Systems

- Interconnected systems with stringent physical constraints to ensure grid balancing
- Long system lifecycles: new solutions built on existing ones (they are not created from scratch)
- Compliance with strict safety and environmental rules
- Compliance with dependability and availability constraints (to ensure security of energy supply)
- Involvement of multiple stakeholders: clients, regulatory authorities, grid operators, energy providers, insurers, urban and land-use planning, plant operators..., with different and possibly contradictory objectives
- Moving context with increasing uncertainties (due to geopolitical tensions, energy market instabilities, climate change, lack of energy policy coordination between countries, evolution of demand wrt. new usages...)

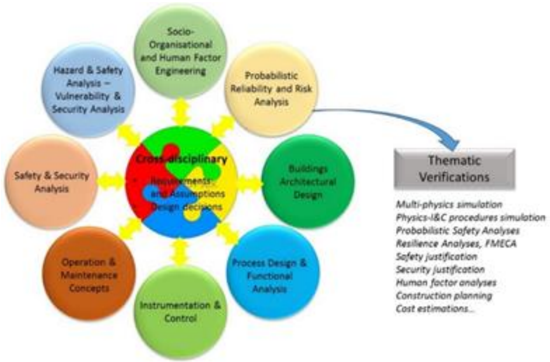
Energy systems are globally over constrained.
New generation of methods & tools are needed to help engineers
find the best compromise for covering multiple "what-if" operational situations (incl. variabilities and hazards)

What Should Be Improved in CPS Engineering?

Today

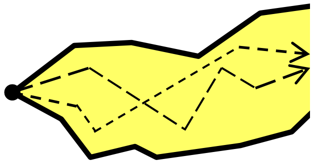
- **system evaluation** is performed mostly with **static models** **(or dynamics are considered too lately)**
- most **verifications are performed manually (or with domain-specific tools)** and hence not as often as necessary
- **information is difficult to share** between disciplinary engineering teams

➔ oversizing, late error detections, and eventually delays and cost overruns



There is a **need for more rigorous engineering method** to

- **Be more effective to assess the impact of each solution** all along the system lifecycle including during preliminary design phases
  → guide and justify design choices also for non-experts
- **Open the solution space to innovative products or services**
  → specify only "what is needed"



Figures: T. Nguyen

CRML A Part of the Solution

Idea =

Use of **realistic dynamic behavioral models** to better handle multi-physics & systems' interactions → **e.g. Modelica**

\+

Use of **formal dynamic requirement models** to automate verifications and evaluate multiple "what-if" scenarios → **CRML**

Article focus

Rationale

- Consideration of "System dynamics" as time may be part of new solutions to cover non-regular situations and hence source of cost reductions
- Formal verifications since for many CPS demonstration that the system operates safely is as important as the design itself

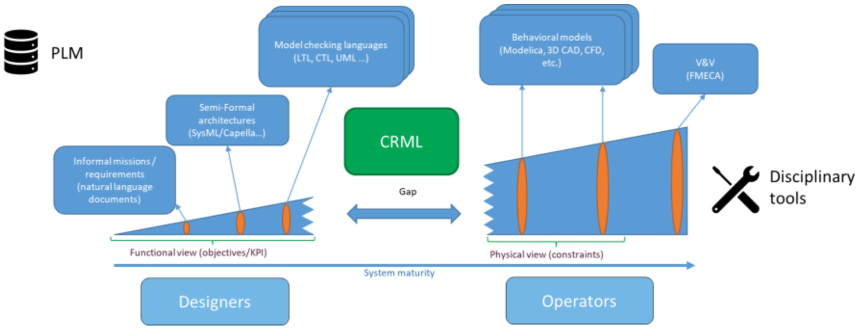
→

Scope of ITEA EMBrACE Project
"An enabler for making the best decisions at each step of the project cycle"

CRML: A Language for Verifying Realistic Dynamic Requirements

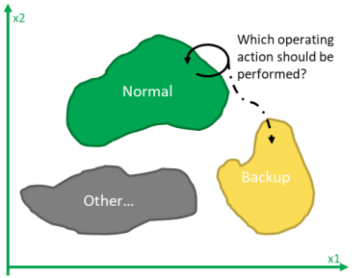
Why a new language?

- Main principles from « System Engineering »
- Tools exists but are incomplete or essentially made for software design
- Native difficulty to address requirements that are « realistic » for systems with strong physical aspects
- In particular to study their dynamical interactions with their environments



CRML positioning vs. State-of-the-Art :
a bridge between the physical & the functional views

A typical realistic dynamical requirement is multiple and stochastic …



1. The system should stay within its normal operating domain.
2. If partial requirement 1 above fails, then the system should go back to its normal operating domain within a given time delay.
3. If partial requirement 2 above fails, or if partial requirement 1 fails with a too high failure rate, then the system should go to a safe backup state within a given time delay.
4. The complete requirement made of the conjunction of partial requirements 1, 2 and 3 should be satisfied with a given probability (e.g., > 99.99%).

… and a typical project quickly sees its complexity increase with the number of requirements/stakeholders and evolution over time

CRML: Not a Whim But a Long-Lasting History

2006

- **EUROSYSLIB project:** start reflections on how to specify systems without describing their detailed behavior ➔ need for a formal specification language
  ➔ investigation of the state-of-the-art.

2009

- **OPENPROD project:** proposition of a link between SysML and Modelica
  ➔ ModelicaML prototype developed by Airbus and tested by EDF.

2012

- **MODRIO project:** proposition by EDF of a new language called FORM-L (Formal Requirement Modelling Language)
  - Specification written by EDF (Thuy Nguyen)
  - Blocks as functions in Modelica
  - Development of two Modelica libraries for the formal capture of requirements: Modelica_Requirement (DLR) and ReqSysPro (EDF).
  - Development of a FORM-L compiler (Inria and Sciworks Technology) on an EDF contract.

2020

- **EMBRACE project:** proposition of CRML as the formal specification of FORM-L.
  - Specification written by EDF (Daniel Bouskela).
  - CRML compiler developed by University of Linköping.

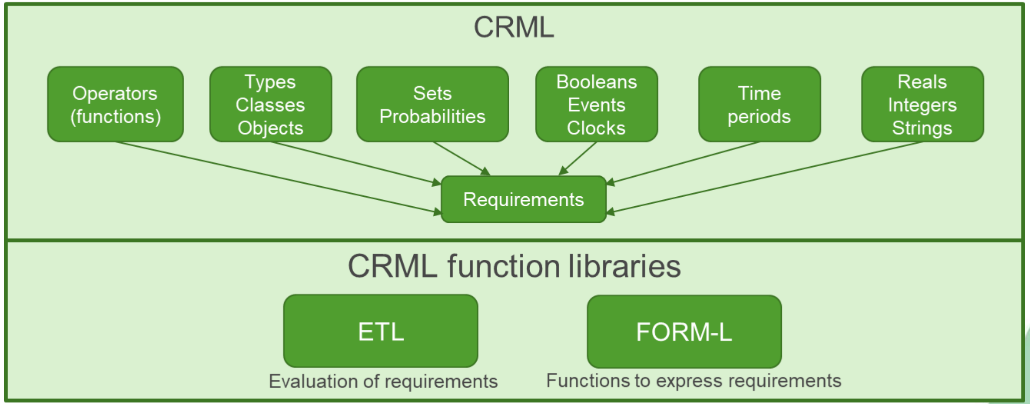
How To Express a CRML Requirement?

R = [Where or Which] [When] [What] + (optional) [How well]

```
for all' pump 'in' system.pumps 'during' system.inOperation 'check count' (pump.isStarted 'becomes true') '<=' 3;
'during' systemOperatingLife 'check at end' (estimator Probability (noStart at inOperation 'becomes false')) '>' 0.99;
```

- Combination of 4 items
  - Spatial locators
  - Time locators
  - Condition to be checked
  - (optionally) Performance indicator

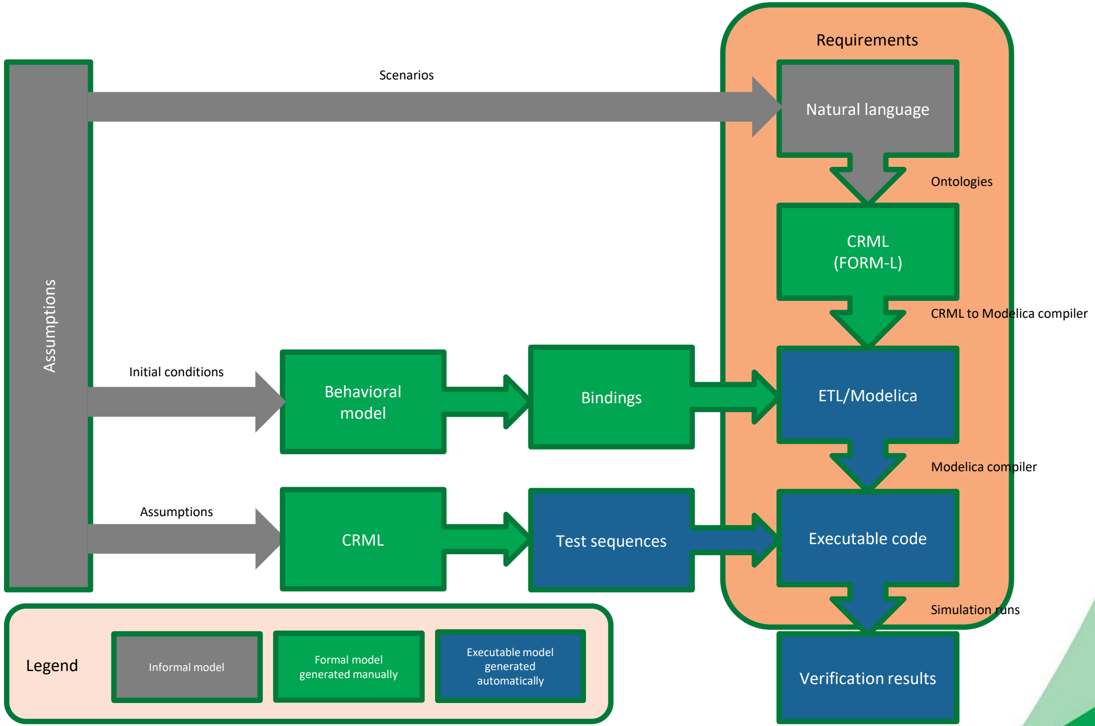
- Value at instant t is a Boolean4 which can be : `true`, `false`, `undefined` or `undecided`



CRML Specification v1.1 (EMBrACE D2.1, Daniel Bouskela)

How to Use CRML for Verifications?

- **Requirement models**
  to capture all constraints on the system and define envelopes of acceptable behaviors
- **Behavioral models**
  to capture the behavior of design solutions
- **Verification models**
  to automate tests
  by using requirement models as observers
  to check whether design solutions meet requirements or not.



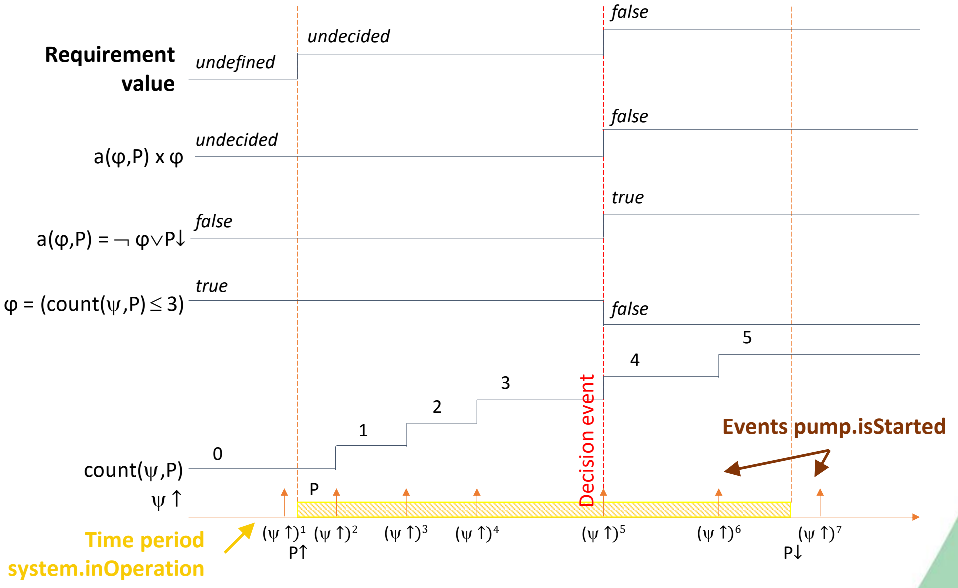
How To Evaluate a CRML Requirement?

Case 1: Requirement R3 is declared as « violated » as soon as condition φ becomes false

Requirement capture in CRML

```
Class Pump is {
        Boolean isStarted is external};
Class System is {
        Pump{} pumps is external;
        Boolean inOperation is external};
System system;

Requirement R3 is {
  'for all' pump 'in' system.pumps
  'during' system.inOperation
  'check count'(pump.isStarted 'becomes true')
  '<=' 3;
};
```



external keyword is used to retrieve values in solution models
Operators in '' are defined by user to improve readability

Requirement evaluation
via observation of system behavioral dynamics

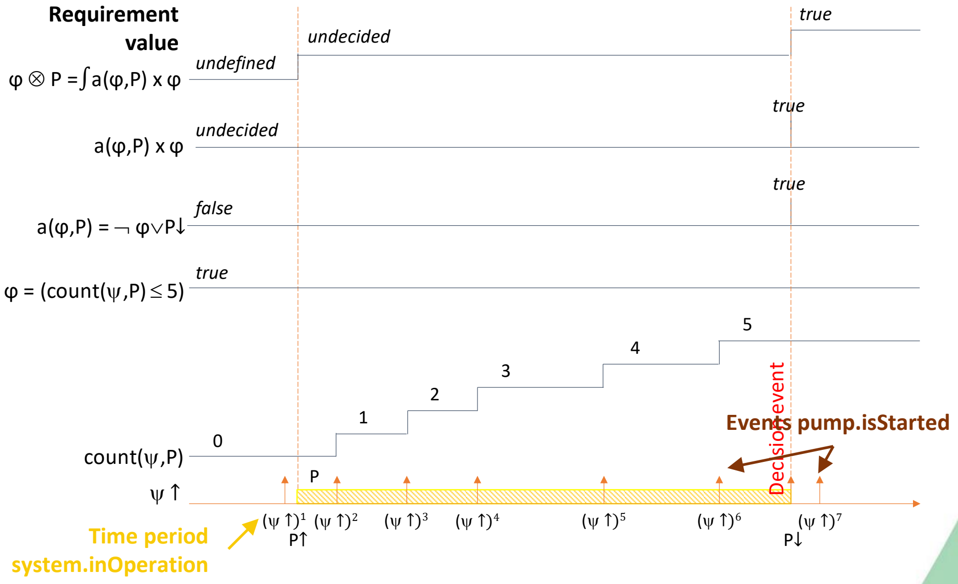
How To Evaluate a CRML Requirement?

Case 2: Requirement R5 is declared as
« undecided » until time period is completed

Requirement capture in CRML

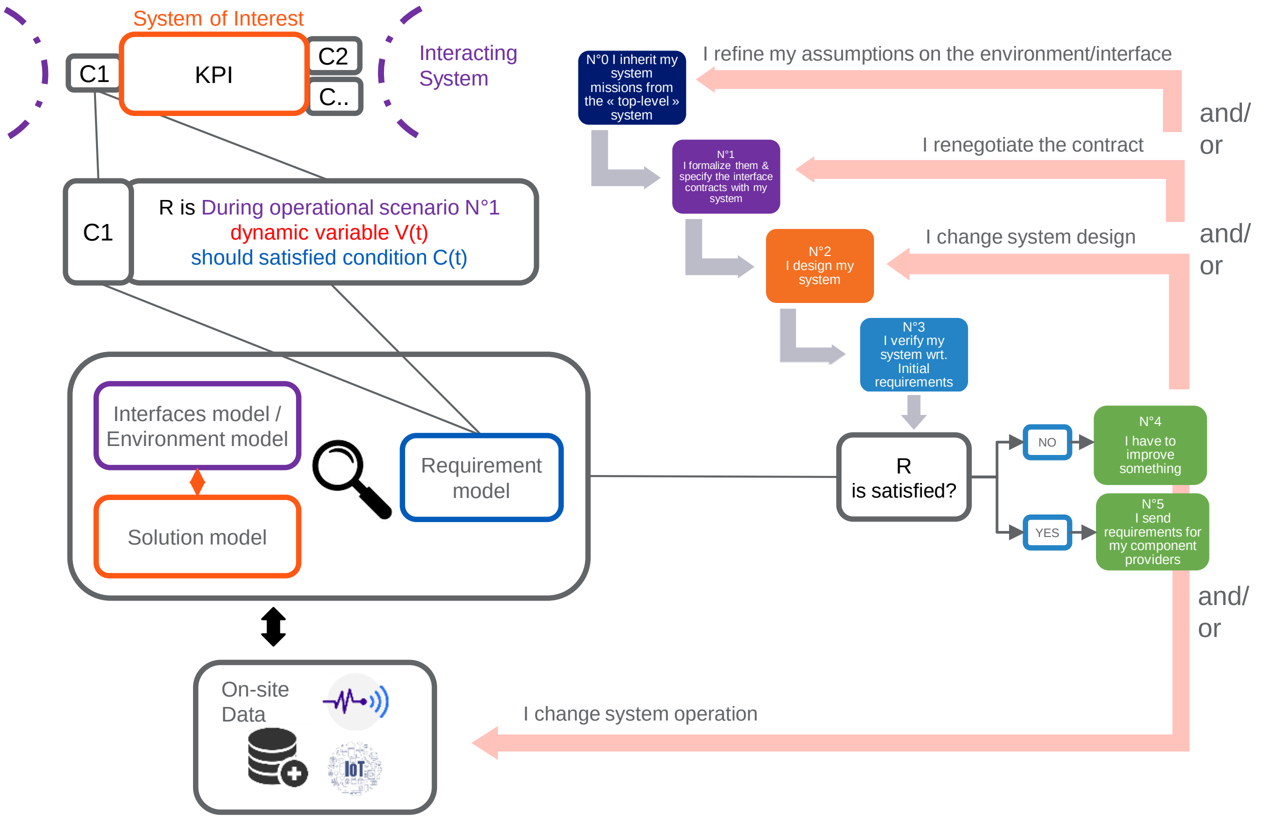
```
Class Pump is {
        Boolean isStarted is external};
Class System is {
        Pump{} pumps is external;
        Boolean inOperation is external};
System system;

Requirement R5 is {
  'for all' pump 'in' system.pumps
  'during' system.inOperation
  'check count'(pump.isStarted 'becomes true')
  '<=' 5;
};
```



Requirement evaluation
via observation of system behavioral dynamics

How to Use CRML As a Decision Tool?



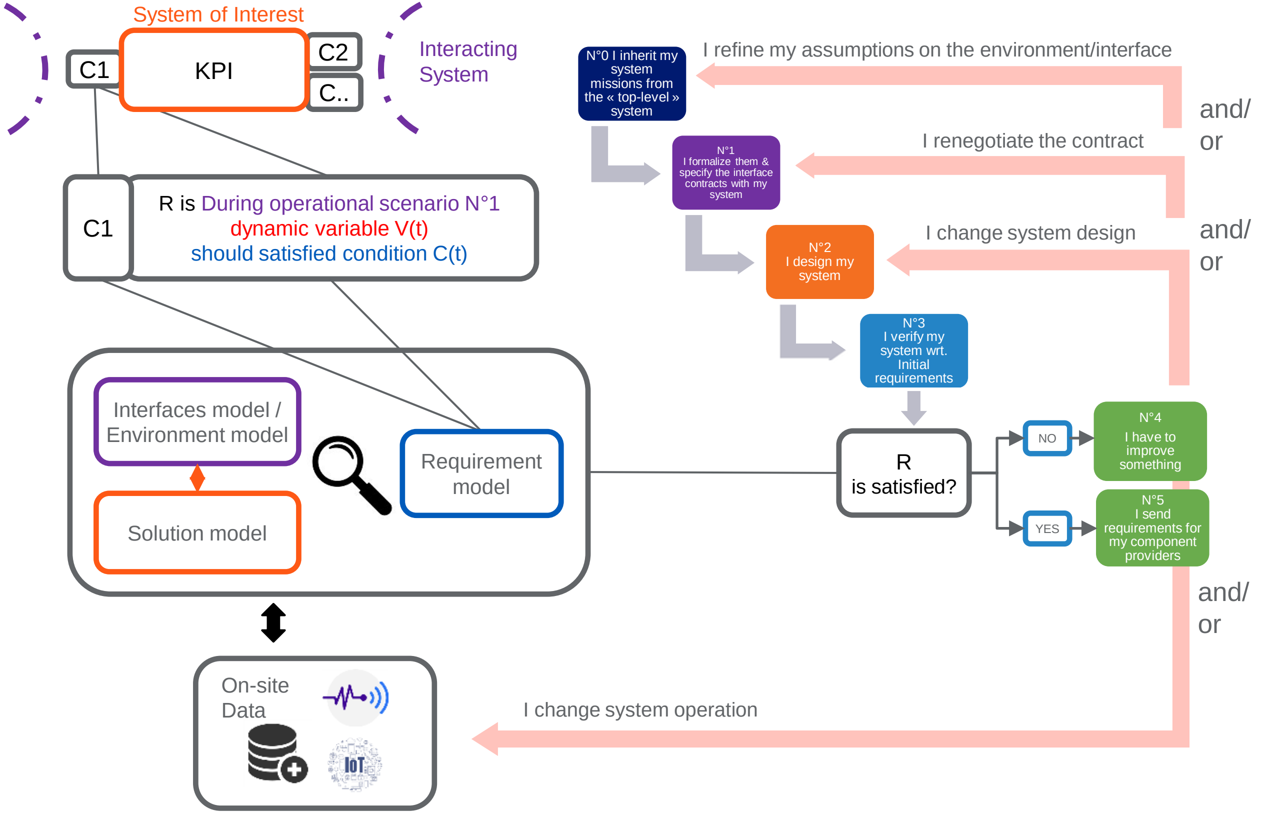
Model to support complexity

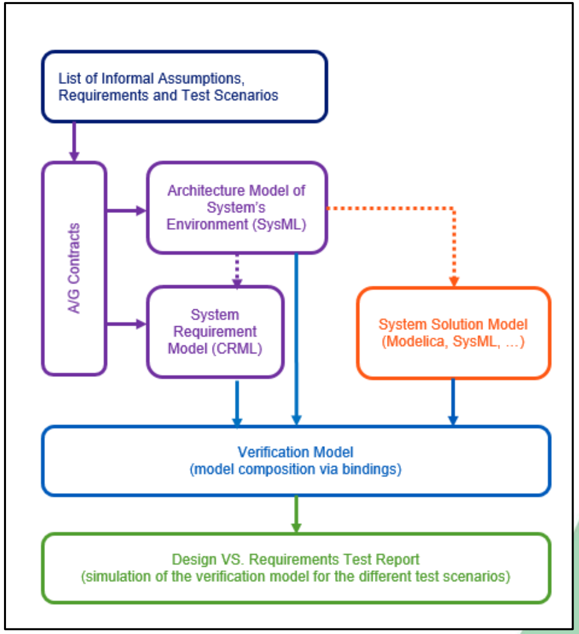
- Scope of responsibility of stakeholders
- Multiplicity of constraints and operating scenarios
- Dynamics of interactions between systems, human and environment

Center development on the requirements

- Evaluate the impact of each solution on your overall ambition
- Design only for the « right » need
- Adapt the studies to « what is just needed »
- All along the project
- And according to the data available at instant T

How to Use CRML As a Decision Tool?

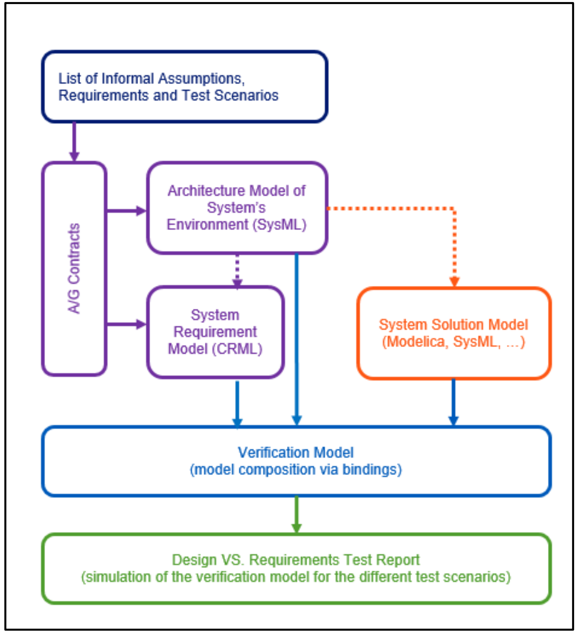




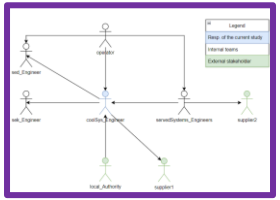
Corresponding modelling architecture

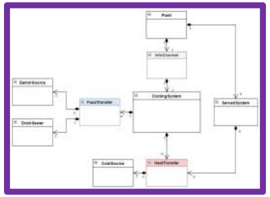
Cooling System Example

Informal System Mission : “Evacuate the heat produced by several served systems with the use of demineralized water at a good availability rate”



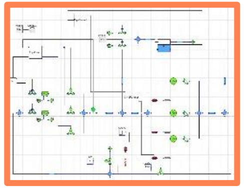
Stakeholder & Architectural Models





CRML Model

ThermoSysPro model

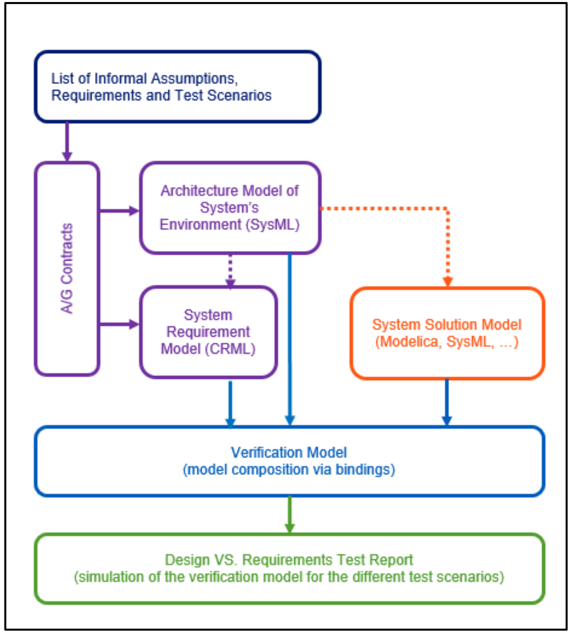


(manual) Bindings

Verification model

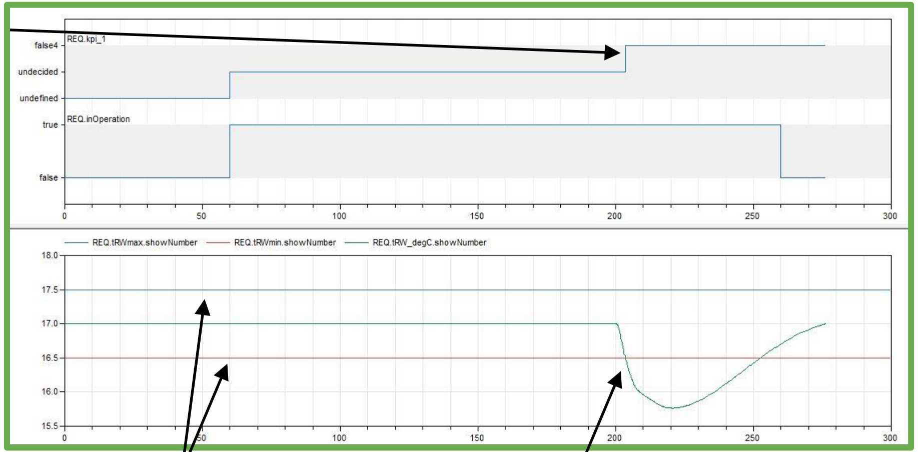
```
Requirement kpi1 is
 'during' inOperation
 'ensure' tRW >= (17 Celsius - tol.)
     and tRW <= (17 Celsius + tol.);
```

Cooling System Example



Simulation results of the verification model

Violation of "kpi1" computed by the CRML model



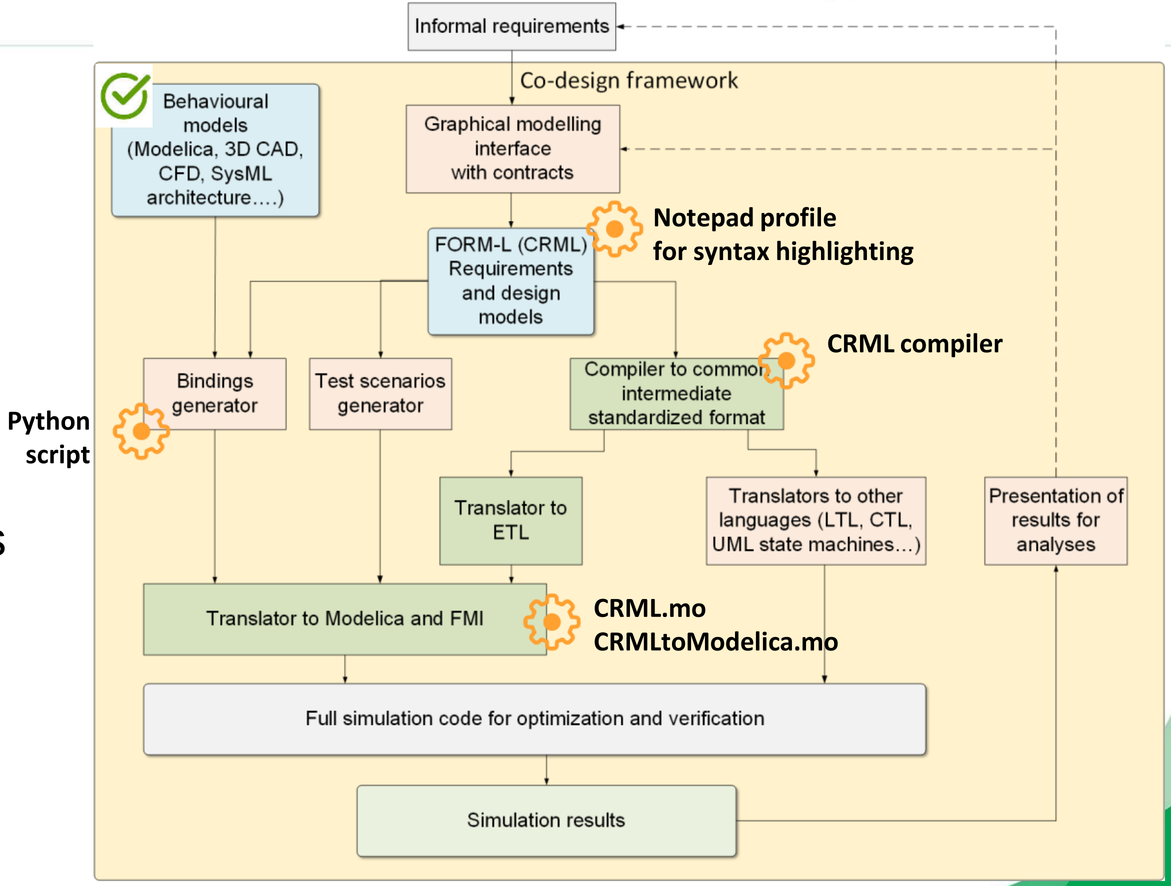
Tolerance interval for temperature

Temperature computed by the ThermoSysPro model

CRML Toolchain

Prototypes to

- Edit CRML
- Compile CRML (to Modelica)
- Simulate CRML
- Build verification models



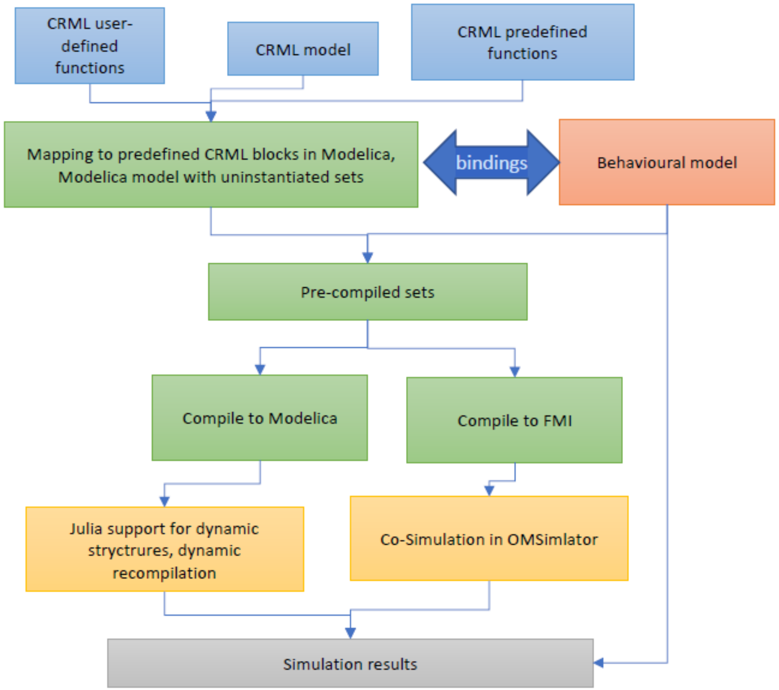
Tools available

Prototypes available

CRML To Modelica Compiler

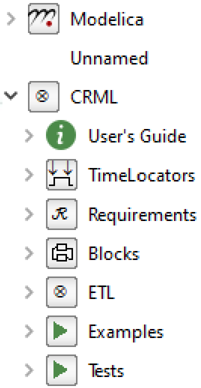
- Developed by Linköping University
- Takes CRML models as input and produces Modelica models as output (based on the CRMLtoModelica.mo library).
- Works only on a subset of CRML (development still ongoing)

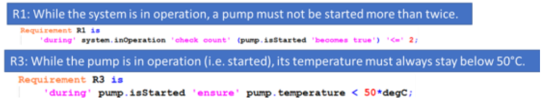
- Available there →



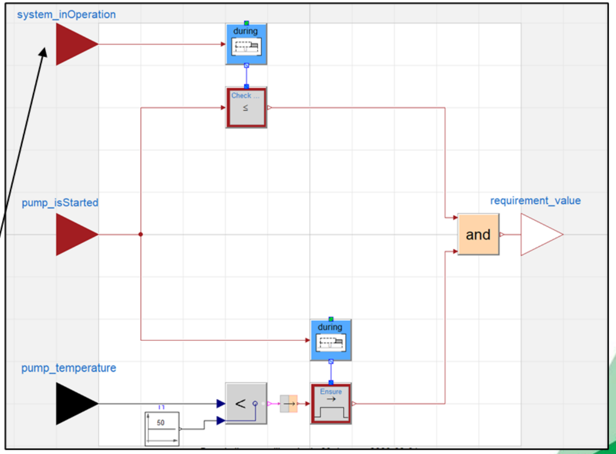
CRML Modelica Library

- Library developed to verify the correctness of Boolean4-based (ETL) algebra for evaluating requirements
- The library contains blocks to express time locators, conditions and probabilities
- Requirements are built by connecting the blocks together in the form of block diagrams
- Available there →





A requirement is built by connecting a time locator block to a condition block



External variables are provided by behavioral model

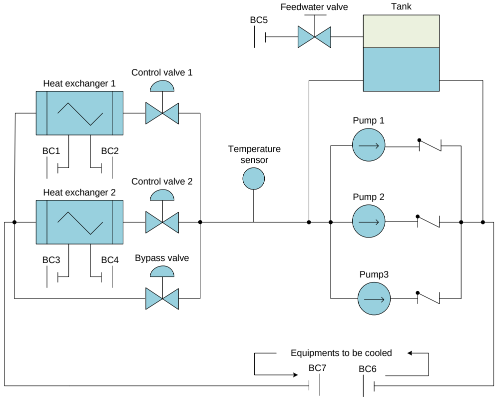
Requirements can be connected together

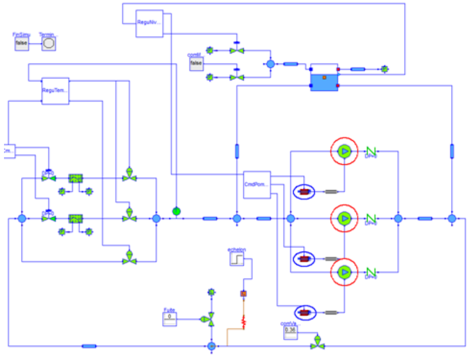
(Semi-)Automation of Verification Models

- Bindings script enable to (semi)automate model composition to build verification models
- Main purpose = evaluate at simulation time the formal requirement expressions that contain quantifiers on **external sets** and conditions on **external variables** whose values is computed in solution models
- Example:

```
Requirement R3 is {
   'for all' pump 'in' system.pumps
   'during' system.inOperation
   'check count'(pump.isStarted 'becomes true')
   '<=' 3;
};
```

pumps is an external set
inOperation and isStarted are external variables





Need to map instances of set "pumps" to elements of the physical model ("Converters" may be needed!)

First Industrial Experiments with CRML (and previous related work called FORM-L)

Evaluation of solution alternatives

- Use of requirement models as an objective comparison criteria

Automation of FMEA studies

- Use of requirement models to define the impact of a faulty component on system's missions and its criticality

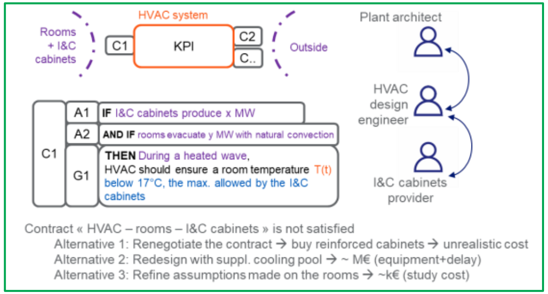
Automation of impact analysis

- Use of requirement models to propagate changes in design assumptions

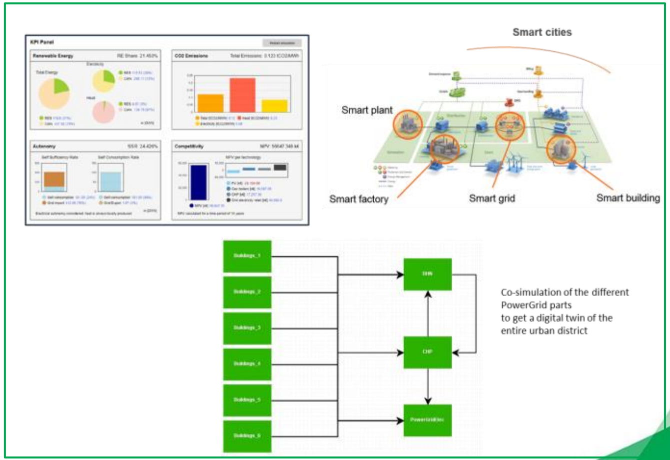
Stakeholder coordination

- Use of requirement models to conceive large-scale systems and to prepare model interfaces for the assembly of their digital twins

…



Solution comparison for HVAC design



Energy renovation of an urban district

CRML at a Glance

A language for **verifying realistic dynamic requirements**
- Accelerate design decisions by evaluating their overall impact
- Automate some repetitive verification tasks
- Concentrate on the design itself and the search for new solutions

A language to **train to “industrial” system design** beyond the physical equations taught in schools

A **language codeveloped** with industrial partners (Saab, Siemens …) and software editors (OpenModelica, TheReuseCompany …)
→ European ITEA project EMBrACE

First support tools to
- Edit CRML model
- Compile CRML
- Simulate CRML (as a Modelica Library)
- Help the construction of verification models (bindings)

→ Open source CRML tutorial

First methodological elements to ease appropriation
- E.Azzouzi’s PhD thesis: stakeholders’ coordination
- B.Mazurié’s Master thesis: graphical guided design method

Industrial context …

Operating constraints
- Specifications
- Variables of interest
- Use case
- Manufacturer limitations
- Norms and Regulations

+

Theory …

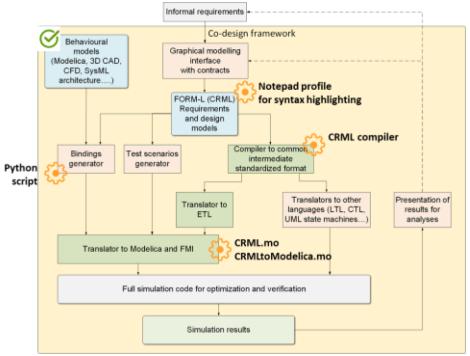
(multi-) Physical System

$$\frac{d(\rho \cdot V)}{dt} = \dot{m}$$

Design assumptions
- Inherited from past decisions (architecture, datasheet, …)
- Issued from project decoupling between domain teams (interfaces, boundary conditions)

=

Case Study



Tools available

Prototypes available

Outlook

- First experiments on energy systems showed that CRML allows to automate some engineering tasks with relatively poor human-added value (e.g., manual evaluation of a change in design assumptions for tens till hundreds of sizing configurations)

- Consequently, more verifications can be performed all along the project and engineers may focus on the solution dynamic behavior to retrieve some margins (as time may be source of cost reductions)

Outlook

- Rethinking historical engineering practices is still a long journey
  but things are changing (adoption of PLM solutions, “textual” requirement databases…)
- CRML can foster such acculturation to System Engineering practices by offering a better integration with disciplinary computation tools → requirements are really used to “relieve the pain” and do not appear anymore as an extra-activity “just” made for documentation

- (Large) Adoption of CRML requires:
  - Communication efforts → with experiments on other industrial cases
  - Maturing of supporting tools
    - (very) small requirement models can be developed with Modelica (CRML.mo) but larger models need CRML compiler → work in the coming months to increase the set of instructions supported and integration into OMEdit
  - Persistent tools / ecosystem → a will to promote CRML as a future standard and a need to reinforce link with SysML v2 to ease adoption by the System Engineering community and to provide graphical patterns for end-users to ‘transparently’ generate CRML

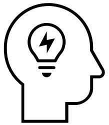
Outlook

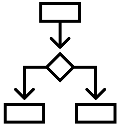
- Towards a new collaborative project on « Sustainable engineering »

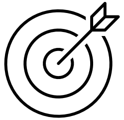
New generation of tools give me a chance an engineer
to check « at my level » that every choice is well informed

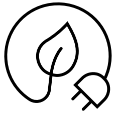
And goes in the direction of more ambitious common objectives at
project level (operational performance)

and society (sobriety, efficiency, resilience, decarbonization)









« Better modelling for better designing just as needed »

ITEA 3

Thank you for your attention!

Contact:

audrey.jardin@edf.fr

lena.buffoni@liu.se

- CRML tutorial
  https://github.com/OpenModelica/CRML/tree/main/resources/crml_tutorial

- CRML specifications
  http://crml-standard.org

- CRML compiler
  https://github.com/OpenModelica/CRML

- ITEA EMBrACE project
  https://itea4.org/project/embrace.html

LINKÖPING UNIVERSITY

eDF