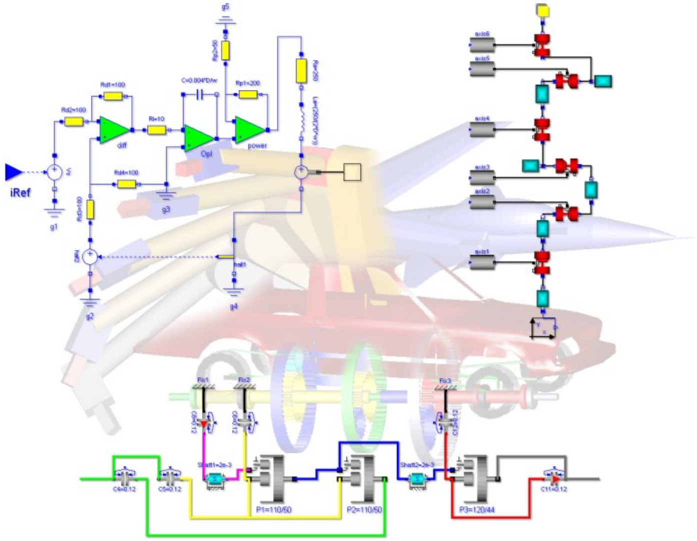
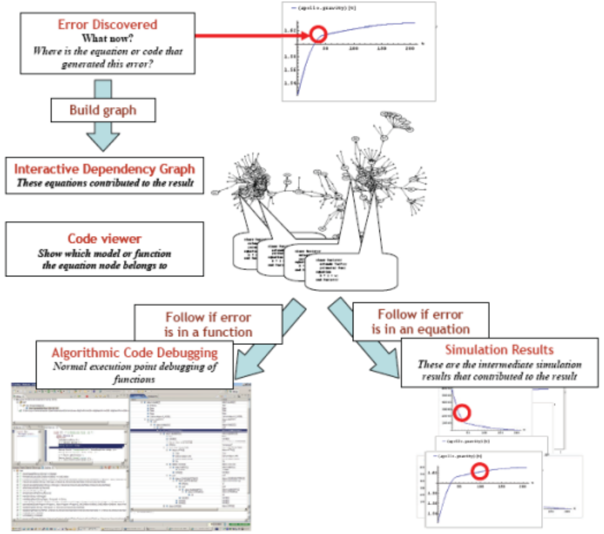
OpenModelica - The Common Requirement Modelling Language (CRML) Integration

Adrian Pop, Lena Buffoni, Audrey Jardin, Daniel Bouskela

2025-02-05

Open Source Modelica Consortium
PELAB, Linköping University
EDF, Électricité de France





Error Discovered
What now?
Where is the equation or code that generated this error?

Build graph

Interactive Dependency Graph
These equations contributed to the result

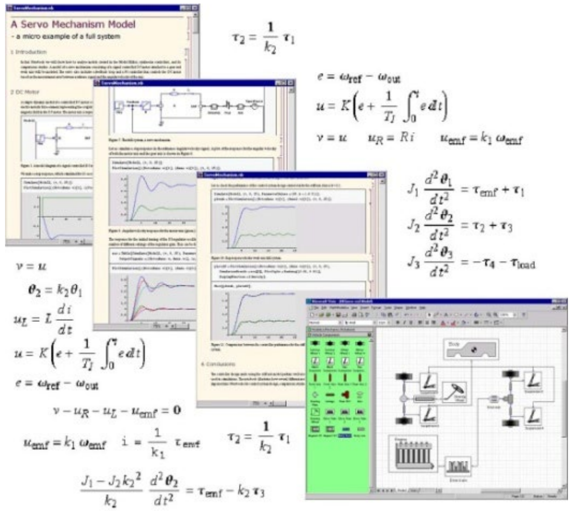
Code viewer
Show which model or function the equation node belongs to

Follow if error is in a function

Follow if error is in an equation

Algorithmic Code Debugging
Normal execution point debugging of functions

Simulation Results
These are the intermediate simulation results that contributed to the result



Outline

- What is CRML
  - The Common Requirement Modelling Language
- CRML Tooling
  - The CRML Compiler
- CRML Integration
  - OMEdit & VSCode & Online
  - Status
- Future work

What is CRML?

- The Common Requirement Modelling Language
  - Language for Verifying Realistic Dynamic Requirements

- Started at EDF around 2006
- Further developed during the ITEA3 EMBrACE project

Ambition: Effective Engineering of Large 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 failure modes, effects, and criticality analysis (FMECA) all along design lifecycle?
- How to justify and document design choices for future generations?

Examples of Challenges - Related to Energy Systems

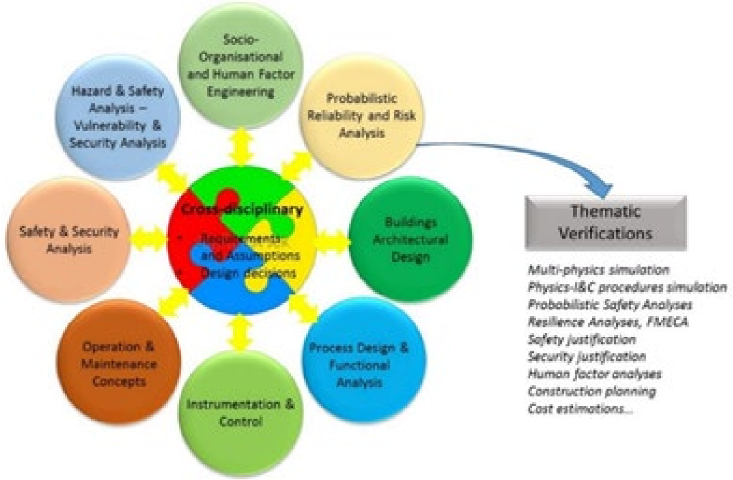
- Interconnected systems with stringent physical constraints to ensure grid balancing
- Long system lifecycles: new solutions built on existing ones (they are not created from scratch)
- Compliance with strict safety and environmental rules
- Compliance with dependability and availability constraints (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)

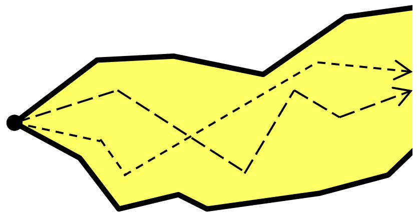
What Should Be Improved in CPS Engineering?

- Today
  - **system evaluation** is performed mostly with **static models** **(or dynamics are considered too late)**
  - 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


Socio-Organisational and Human Factor Engineering
Probabilistic Reliability and Risk Analysis
Hazard & Safety Analysis – Vulnerability & Security Analysis
Safety & Security Analysis
Cross-disciplinary
Requirements and Assumptions
Design decisions
Buildings Architectural Design
Operation & Maintenance Concepts
Instrumentation & Control
Process Design & Functional Analysis
Thematic Verifications
Multi-physics simulation
Physics-I&C procedures simulation
Probabilistic Safety Analyses
Resilience Analyses, FMECA
Safety justification
Security justification
Human factor analyses
Construction planning
Cost estimations...

- There is a **need for more rigorous engineering method** to
  - **Be more effective assessing 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**

Rationale

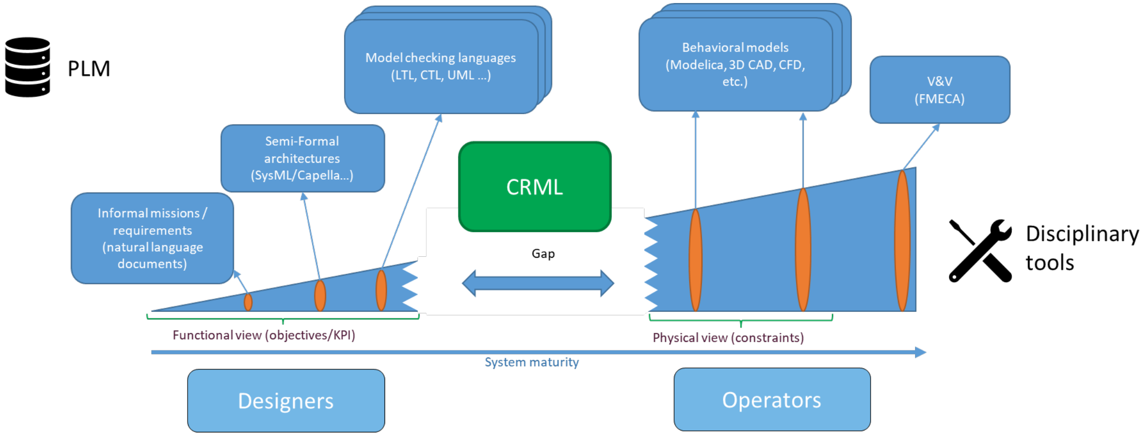
- Consideration of “System Dynamics” as time may be part of new solutions to cover non-regular situations and hence source of cost reductions
- Formal verifications since for many CPS demonstration that the system operates safely is as important as the design itself

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

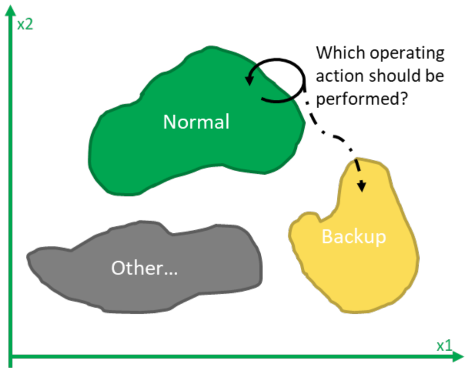
Why a new language?

- 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


PLM
Model checking languages (LTL, CTL, UML ...)
Behavioral models (Modelica, 3D CAD, CFD, etc.)
V&V (FMECA)
Semi-Formal architectures (SysML/Capella...)
Informal missions / requirements (natural language documents)
CRML
Gap
Disciplinary tools
Functional view (objectives/KPI)
Physical view (constraints)
System maturity
Designers
Operators

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

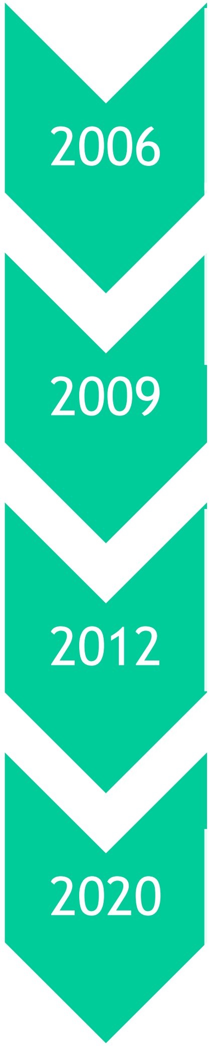
A typical realistic dynamical requirement is multiple and stochastic …



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: a Long-Lasting History



- **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.

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

- **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.

- **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.

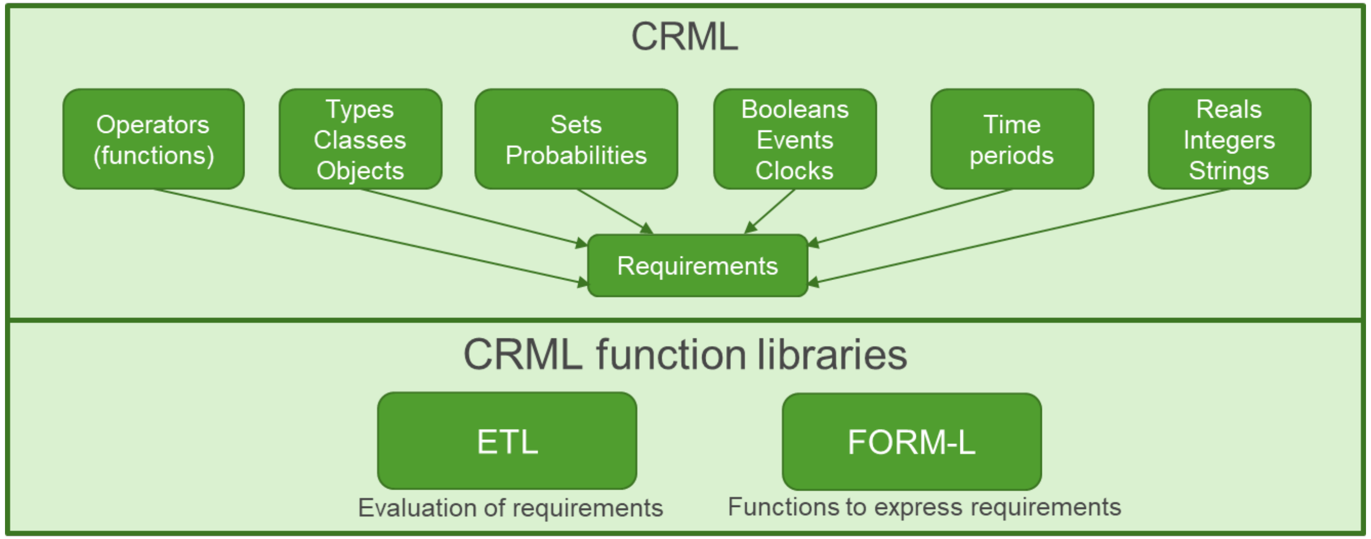
BASEECS 2025 - ITEA5 project, FPP phase
Behavioral Analysis and Simulation for Environmentally and Economically-sustainable Co-Engineered Systems

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;
```

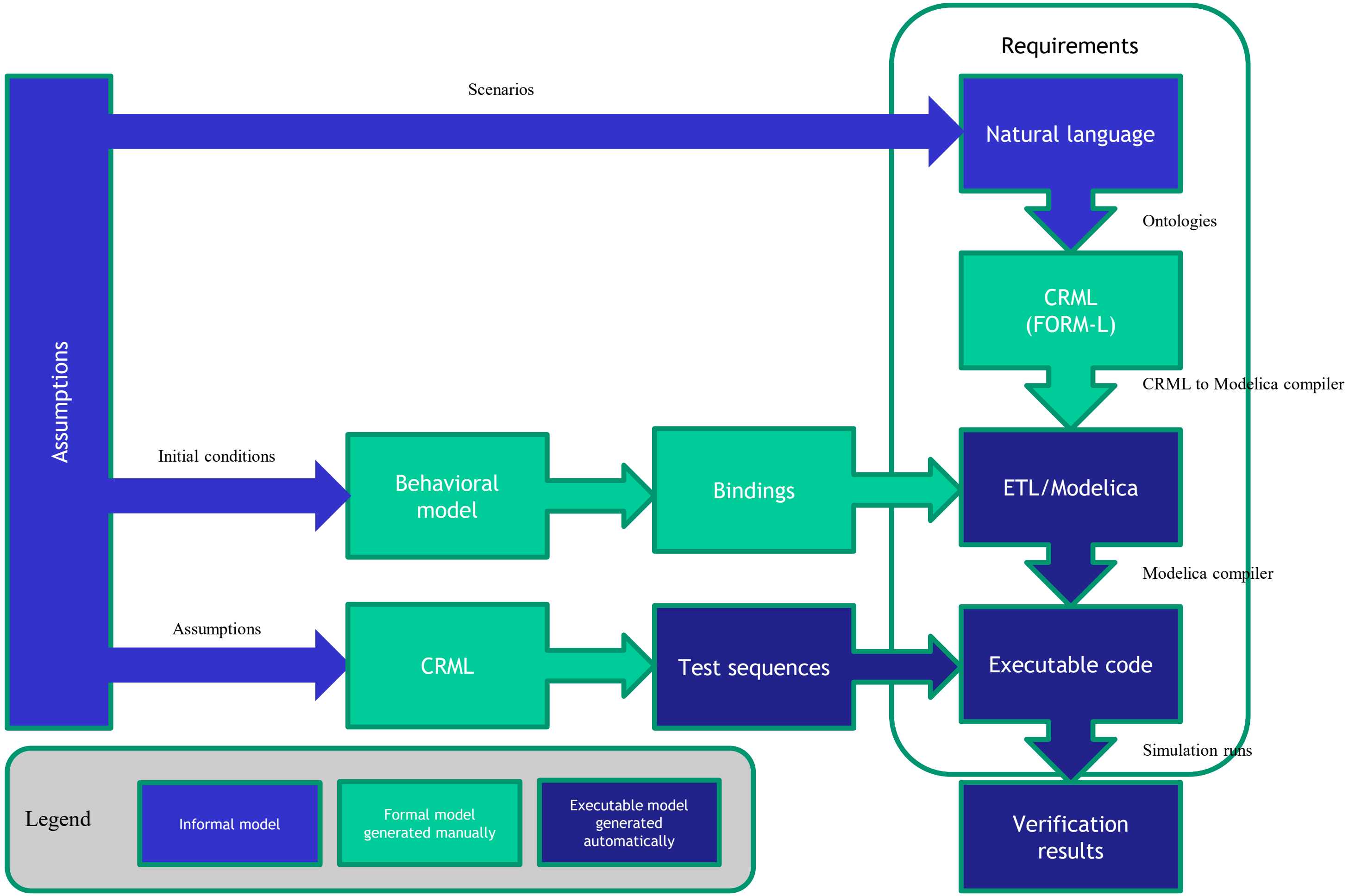
- Combination of 4 items
  - Spatial locators
  - Time locators
  - Condition to be checked
  - (optionally) Performance indicator
- Value at instant t is a Boolean4 which can be : `true`, `false`, `undefined` or `undecided`



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?

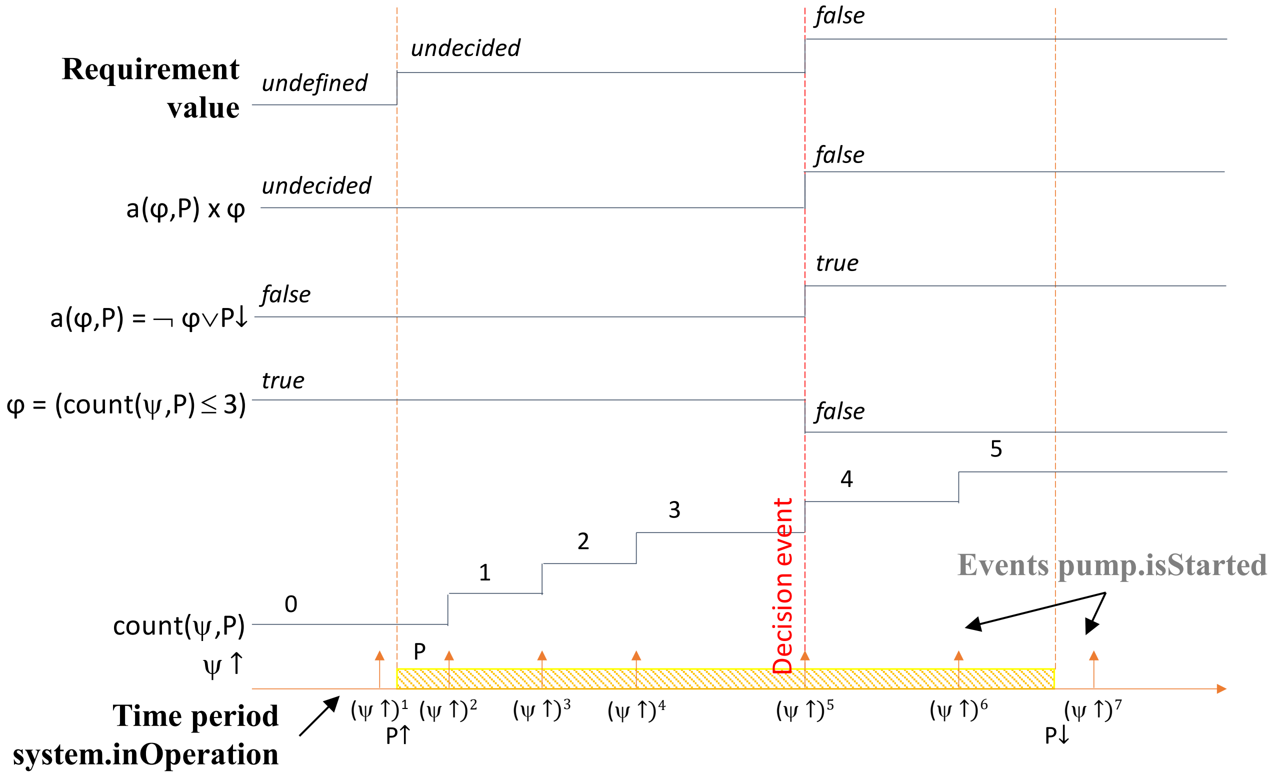
Case 1: Requirement R3 is declared as « violated » as soon as condition φ becomes false

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

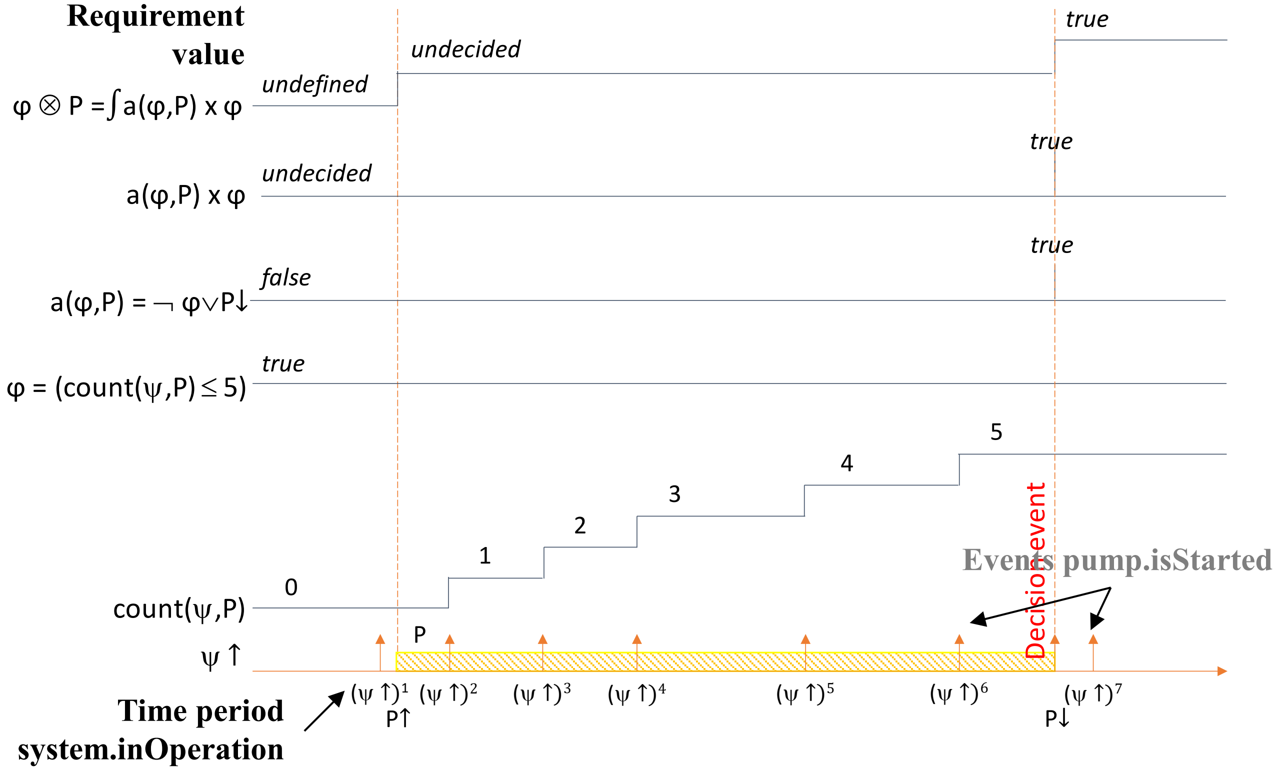
How To Evaluate a CRML Requirement?

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

Requirement capture in CRML

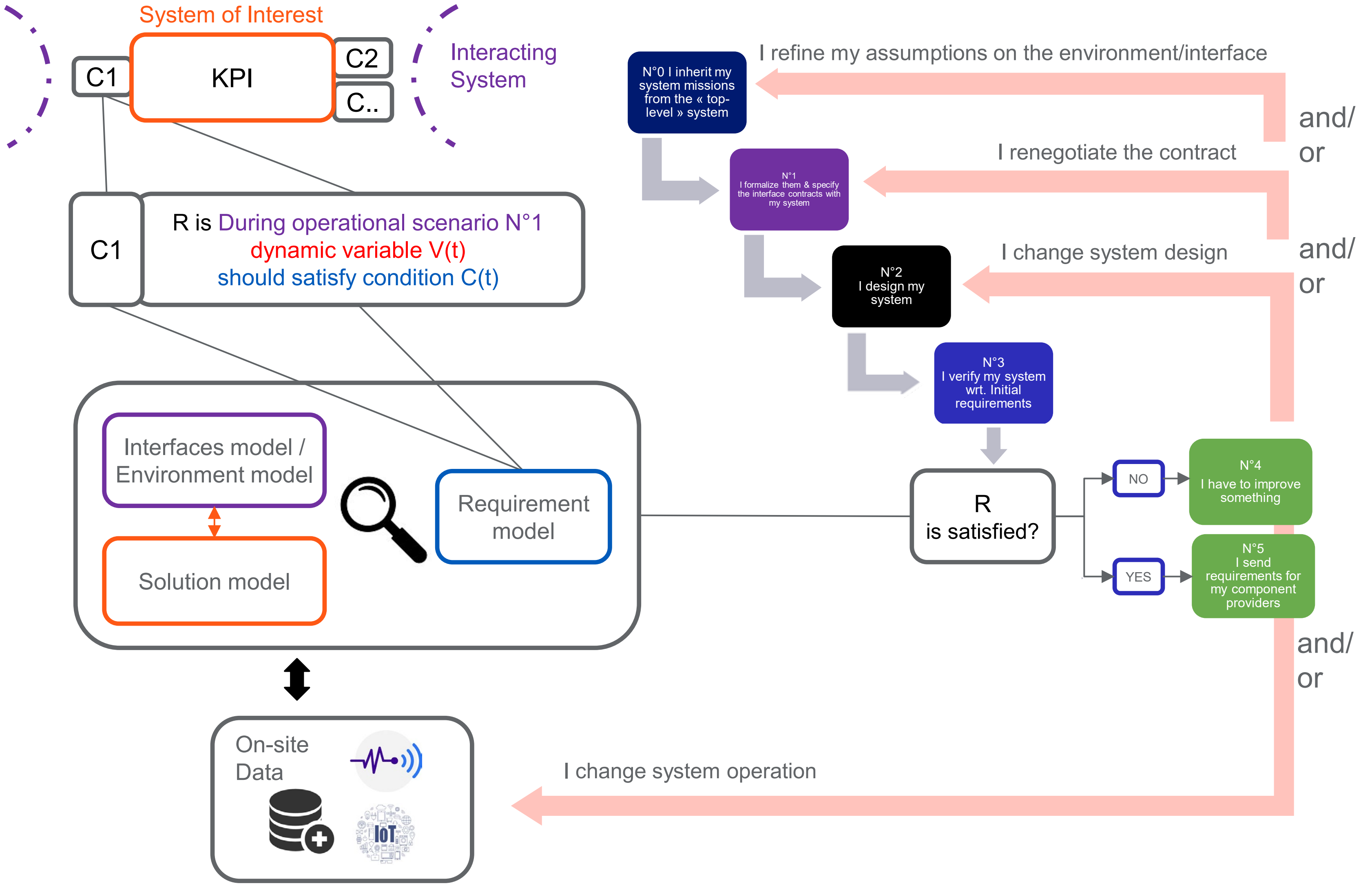
```
Class Pump is {
        Boolean isStarted is external};
Class System is {
        Pump{} pumps is external;
        Boolean inOperation is external};
System system;

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?



System of Interest
KPI
C1
C2
C..
Interacting System
C1
R is During operational scenario N°1
dynamic variable V(t)
should satisfy condition C(t)
Interfaces model / Environment model
Solution model
Requirement model
On-site Data
IoT
N°0 I inherit my system missions from the « top-level » system
N°1 I formalize them & specify the interface contracts with my system
N°2 I design my system
N°3 I verify my system wrt. Initial requirements
R is satisfied?
NO
YES
N°4 I have to improve something
N°5 I send requirements for my component providers
I refine my assumptions on the environment/interface
I renegotiate the contract
I change system design
I change system operation
and/or
and/or
and/or

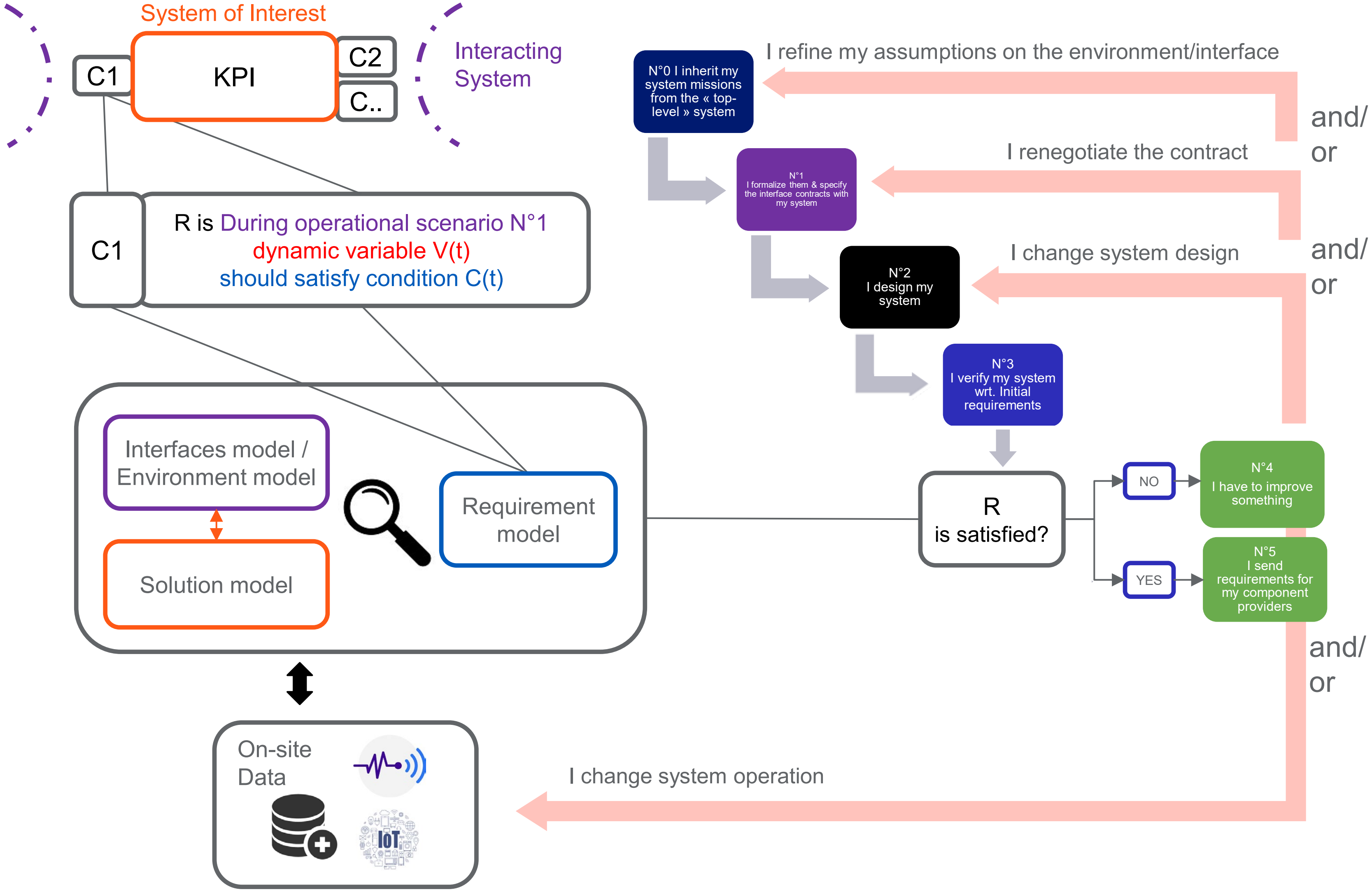
Model to support complexity

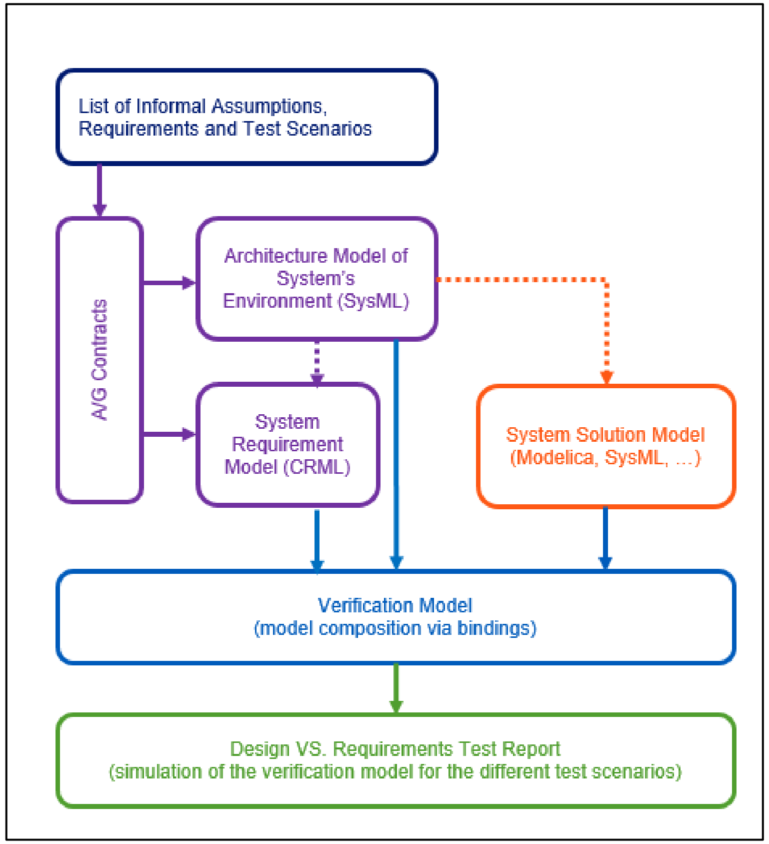
- Scope of responsibility of stakeholders
- Multiplicity of constraints and operating scenarios
- Dynamics of interactions between systems, human and environment

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?


System of Interest
KPI
C1
C2
C..
Interacting System
C1
R is During operational scenario N°1
dynamic variable V(t)
should satisfy condition C(t)
Interfaces model / Environment model
Solution model
Requirement model
On-site Data
IoT
N°0 I inherit my system missions from the « top-level » system
N°1
I formalize them & specify the interface contracts with my system
N°2
I design my system
N°3
I verify my system wrt. Initial requirements
R
is satisfied?
NO
YES
N°4
I have to improve something
N°5
I send requirements for my component providers
I refine my assumptions on the environment/interface
I renegotiate the contract
I change system design
I change system operation
and/
or
and/
or
and/
or


List of Informal Assumptions, Requirements and Test Scenarios
A/G Contracts
Architecture Model of System's Environment (SysML)
System Requirement Model (CRML)
System Solution Model (Modelica, SysML, …)
Verification Model
(model composition via bindings)
Design VS. Requirements Test Report
(simulation of the verification model for the different test scenarios)

Corresponding modelling architecture

Outline

- What is CRML
  - The Common Requirement Modelling Language
- CRML Tooling
  - The CRML Compiler
- CRML Integration
  - OMEdit & VSCode & Online
  - Status
- Future work

CRML Tooling

- The CRML compiler
  - https://github.com/lenaRB/crml-compiler/
  - Implemented in Java
  - Translates CRML to Modelica
  - Integrates with Unit testing and Reporting

- Ongoing work
  - Support the full CRML specification
  - Graphical notation for CRML and support in OpenModelica GUI
  - Continue to improve the integration with OpenModelica GUI

CRML – OMEdit Integration

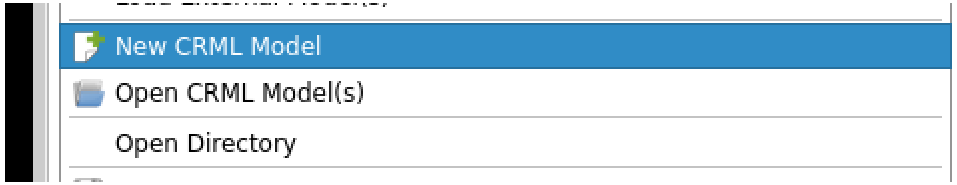
- CRML menus in OMEdit activated by Tools->Options->General->Enable CRML support
  - Generate and load Modelica code (also via the Library/File browser, right click)
    - Call the CRML compiler on the opened CRML file, generate Modelica code, load it into OMEdit, give errors if the code cannot be loaded
  - Dialog for CRML configuration before compilation
    - Set the name of the generated Modelica file, the package name, etc
    - Future
      - annotation in the CRML file where one can provide a configuration
      - Modelica annotation in the generated Modelica file

CRML – OMEdit Integration

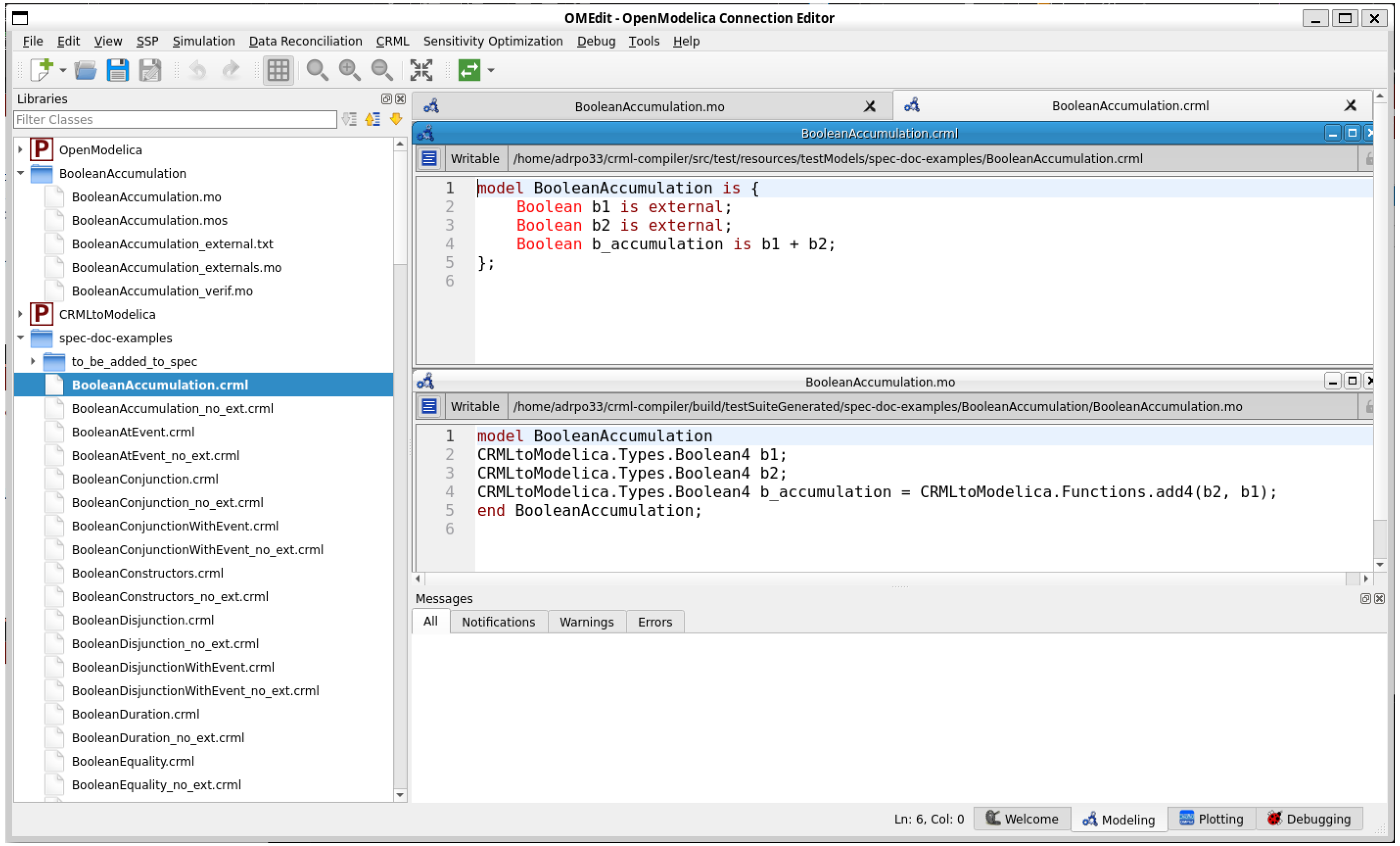
- Run test suite
  - Select a directory with CRML files
  - Call the CRML tool to generate the html report
  - Load and display the html test report
  - A CRML test will go through these phases
    - Parsing
    - Translation
    - Verification model generation
    - Execution
    - Result Verification

CRML – OMEdit Integration

- New / Open CRML models



- Load directories containing CRML models
- Syntax Highlighting


OMEdit - OpenModelica Connection Editor
File Edit View SSP Simulation Data Reconciliation CRML Sensitivity Optimization Debug Tools Help
Libraries
Filter Classes
OpenModelica
BooleanAccumulation
BooleanAccumulation.mo
BooleanAccumulation.mos
BooleanAccumulation_external.txt
BooleanAccumulation_externals.mo
BooleanAccumulation_verif.mo
CRMLtoModelica
spec-doc-examples
to_be_added_to_spec
BooleanAccumulation.crml
BooleanAccumulation_no_ext.crml
BooleanAtEvent.crml
BooleanAtEvent_no_ext.crml
BooleanConjunction.crml
BooleanConjunction_no_ext.crml
BooleanConjunctionWithEvent.crml
BooleanConjunctionWithEvent_no_ext.crml
BooleanConstructors.crml
BooleanConstructors_no_ext.crml
BooleanDisjunction.crml
BooleanDisjunction_no_ext.crml
BooleanDisjunctionWithEvent.crml
BooleanDisjunctionWithEvent_no_ext.crml
BooleanDuration.crml
BooleanDuration_no_ext.crml
BooleanEquality.crml
BooleanEquality_no_ext.crml
BooleanAccumulation.mo
BooleanAccumulation.crml
Messages
All Notifications Warnings Errors
Ln: 6, Col: 0
Welcome Modeling Plotting Debugging

BooleanAccumulation.crml

Writable /home/adrpo33/crml-compiler/src/test/resources/testModels/spec-doc-examples/BooleanAccumulation.crml

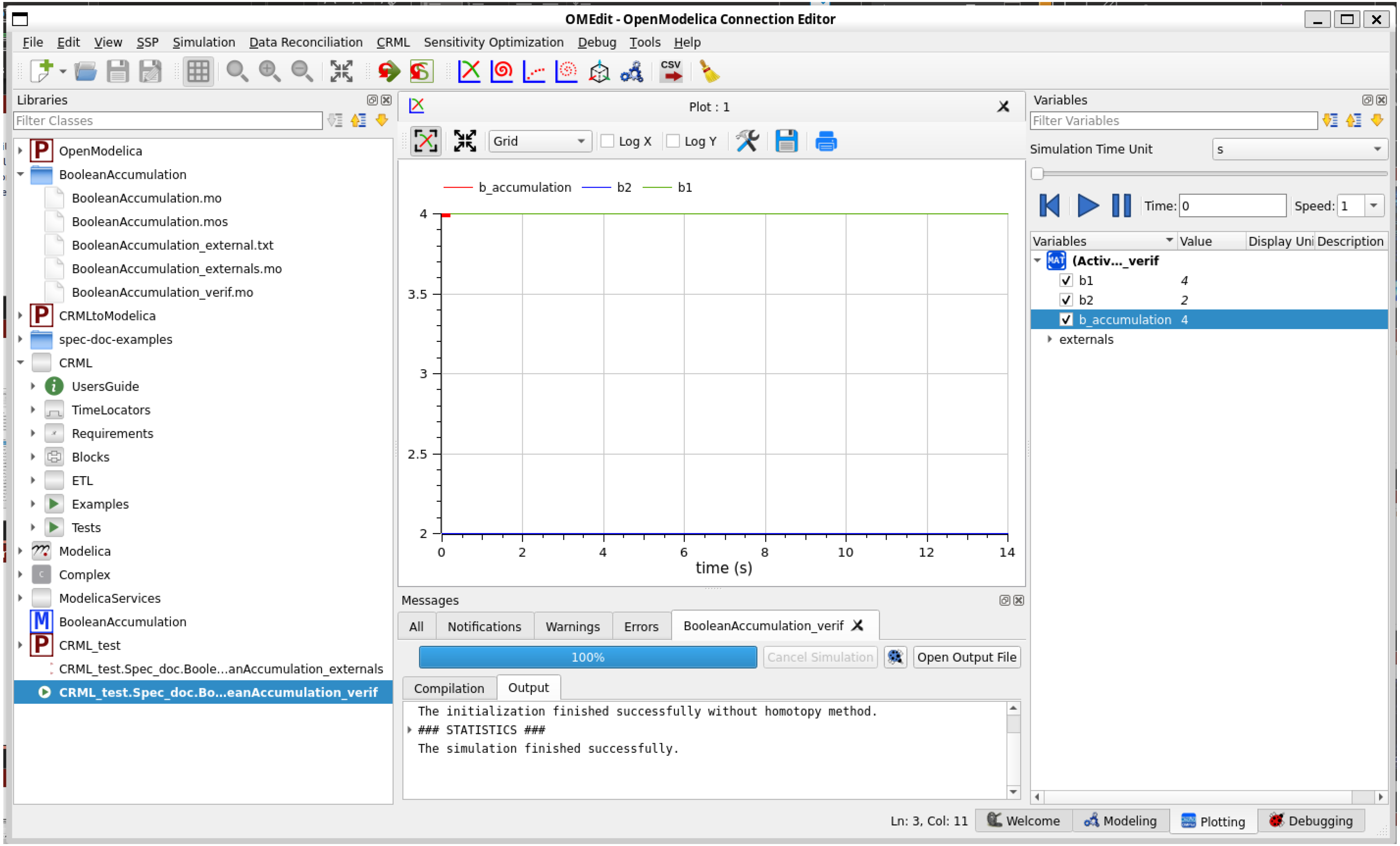
```
model BooleanAccumulation is {
    Boolean b1 is external;
    Boolean b2 is external;
    Boolean b_accumulation is b1 + b2;
};
```

BooleanAccumulation.mo

Writable /home/adrpo33/crml-compiler/build/testSuiteGenerated/spec-doc-examples/BooleanAccumulation/BooleanAccumulation.mo

```
model BooleanAccumulation
CRMLtoModelica.Types.Boolean4 b1;
CRMLtoModelica.Types.Boolean4 b2;
CRMLtoModelica.Types.Boolean4 b_accumulation = CRMLtoModelica.Functions.add4(b2, b1);
end BooleanAccumulation;
```

Generate and Simulate Modelica code

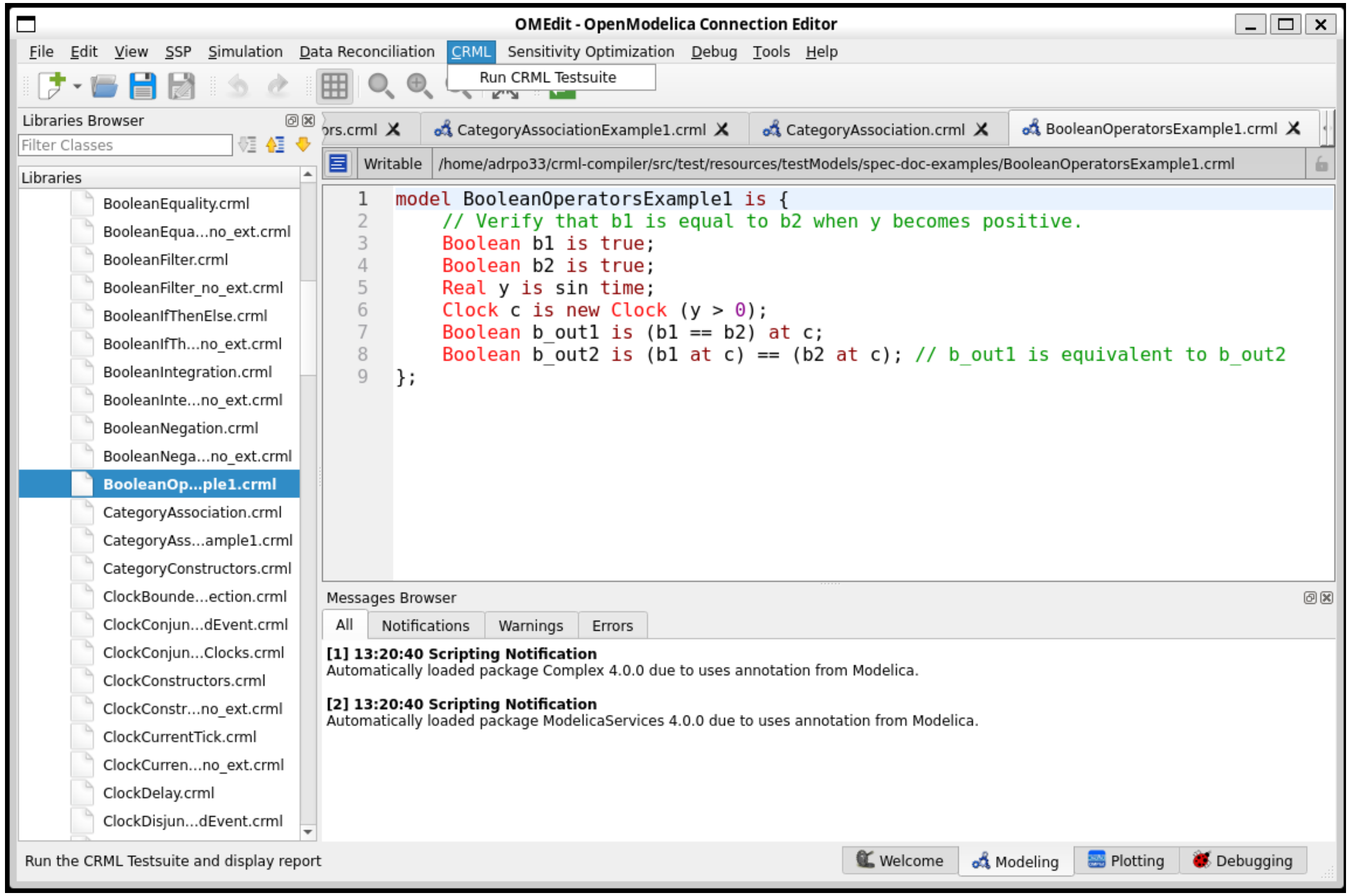


Variables	Value	Display Uni	Description
(Activ..._verif			
☑ b1	4		
☑ b2	2		
☑ b_accumulation	4		
externals			

```
The initialization finished successfully without homotopy method.
### STATISTICS ###
The simulation finished successfully.
```

CRML – OMEdit Integration

- Run CRML Testsuite



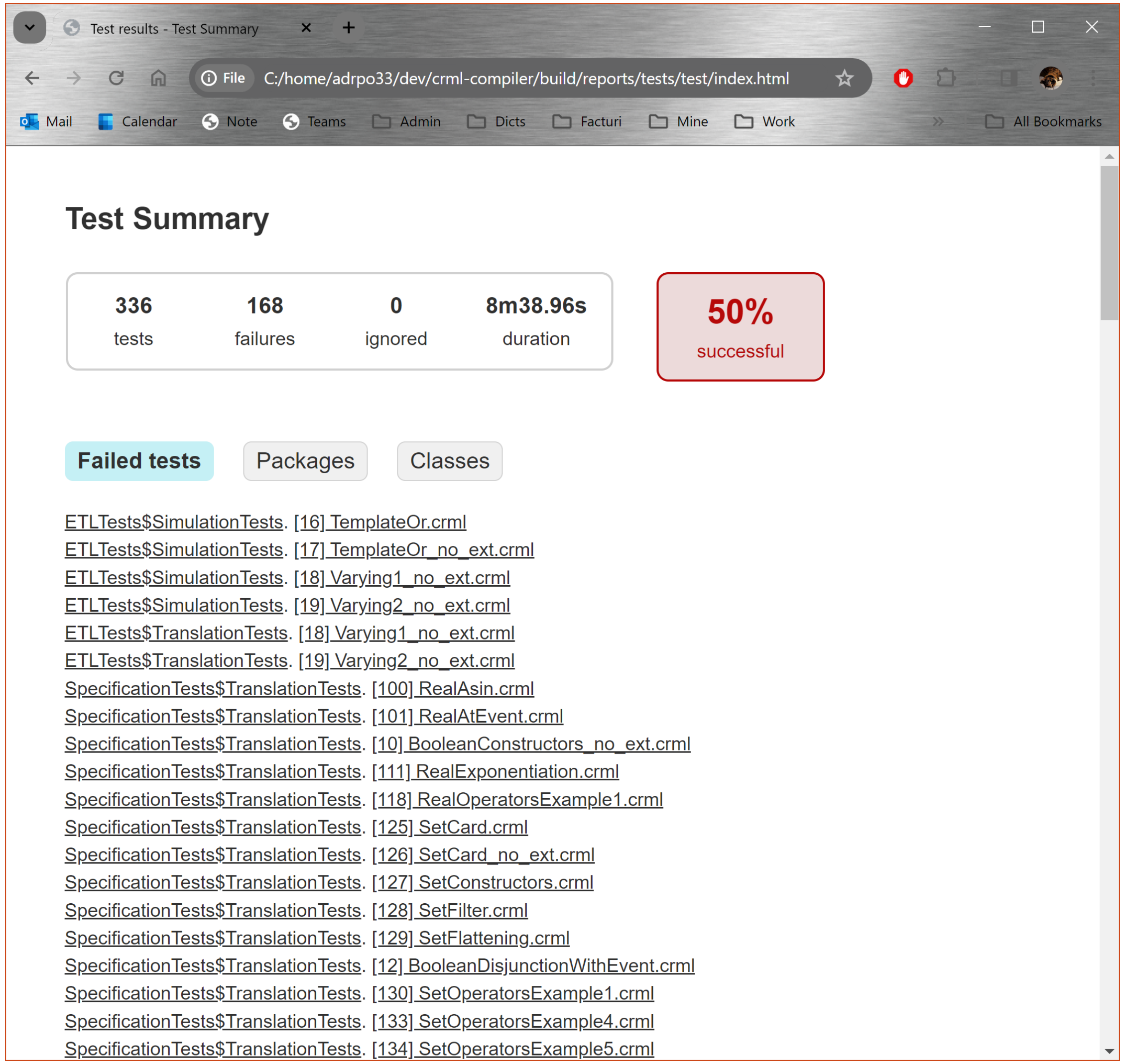
/home/adrpo33/crml-compiler/src/test/resources/testModels/spec-doc-examples/BooleanOperatorsExample1.crml

```
model BooleanOperatorsExample1 is {
    // Verify that b1 is equal to b2 when y becomes positive.
    Boolean b1 is true;
    Boolean b2 is true;
    Real y is sin time;
    Clock c is new Clock (y > 0);
    Boolean b_out1 is (b1 == b2) at c;
    Boolean b_out2 is (b1 at c) == (b2 at c); // b_out1 is equivalent to b_out2
};
```

[1] 13:20:40 Scripting Notification
Automatically loaded package Complex 4.0.0 due to uses annotation from Modelica.

[2] 13:20:40 Scripting Notification
Automatically loaded package ModelicaServices 4.0.0 due to uses annotation from Modelica.

Run the CRML Testsuite and display report


Test results - Test Summary
File
C:/home/adrpo33/dev/crml-compiler/build/reports/tests/test/index.html
Mail
Calendar
Note
Teams
Admin
Dicts
Facturi
Mine
Work
All Bookmarks

Test Summary

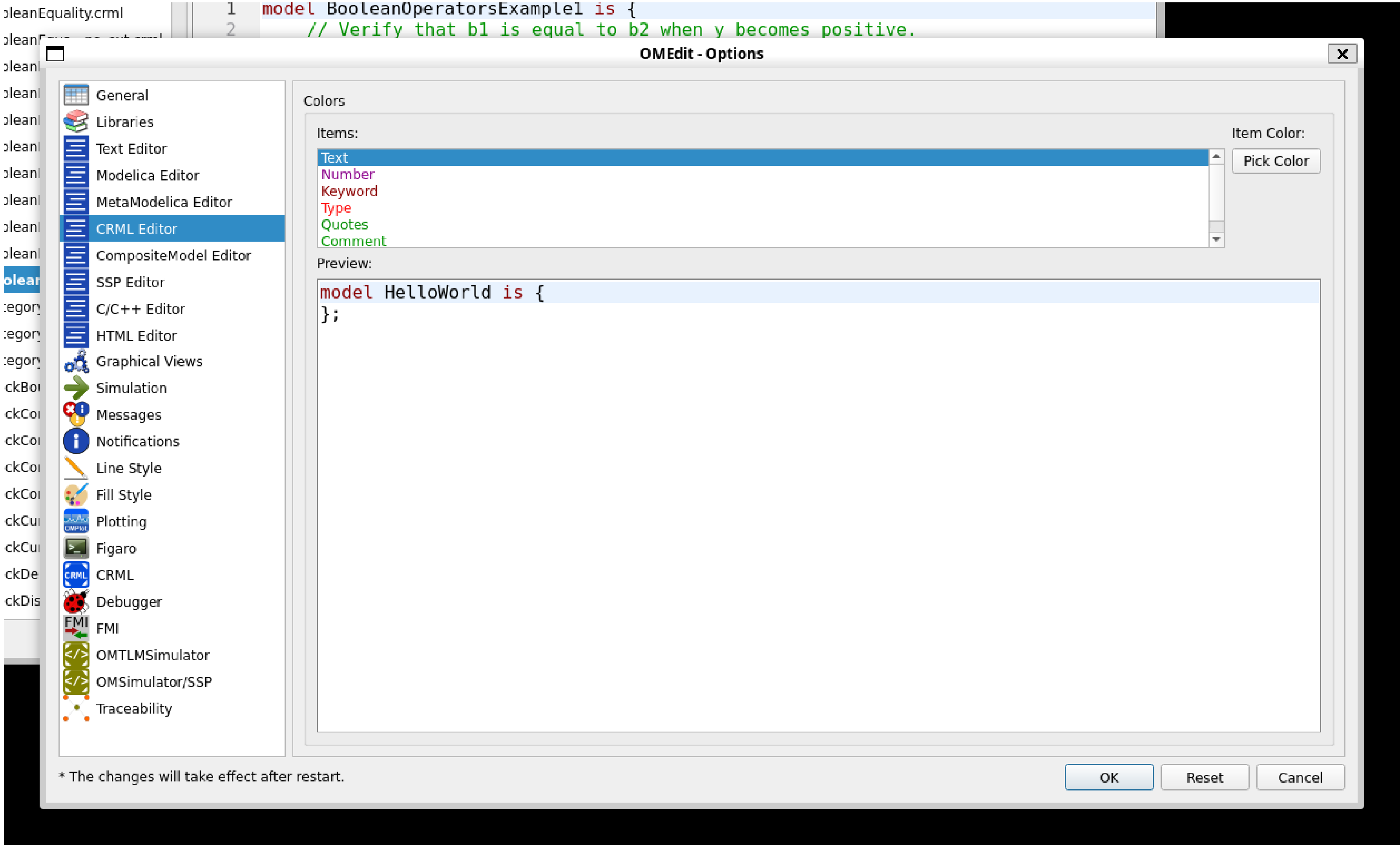
336	168	0	8m38.96s	50%
tests	failures	ignored	duration	successful

Failed tests | Packages | Classes

ETLTests$SimulationTests. [16] TemplateOr.crml
ETLTests$SimulationTests. [17] TemplateOr_no_ext.crml
ETLTests$SimulationTests. [18] Varying1_no_ext.crml
ETLTests$SimulationTests. [19] Varying2_no_ext.crml
ETLTests$TranslationTests. [18] Varying1_no_ext.crml
ETLTests$TranslationTests. [19] Varying2_no_ext.crml
SpecificationTests$TranslationTests. [100] RealAsin.crml
SpecificationTests$TranslationTests. [101] RealAtEvent.crml
SpecificationTests$TranslationTests. [10] BooleanConstructors_no_ext.crml
SpecificationTests$TranslationTests. [111] RealExponentiation.crml
SpecificationTests$TranslationTests. [118] RealOperatorsExample1.crml
SpecificationTests$TranslationTests. [125] SetCard.crml
SpecificationTests$TranslationTests. [126] SetCard_no_ext.crml
SpecificationTests$TranslationTests. [127] SetConstructors.crml
SpecificationTests$TranslationTests. [128] SetFilter.crml
SpecificationTests$TranslationTests. [129] SetFlattening.crml
SpecificationTests$TranslationTests. [12] BooleanDisjunctionWithEvent.crml
SpecificationTests$TranslationTests. [130] SetOperatorsExample1.crml
SpecificationTests$TranslationTests. [133] SetOperatorsExample4.crml
SpecificationTests$TranslationTests. [134] SetOperatorsExample5.crml

CRML – OMEdit Integration

- Editor Settings



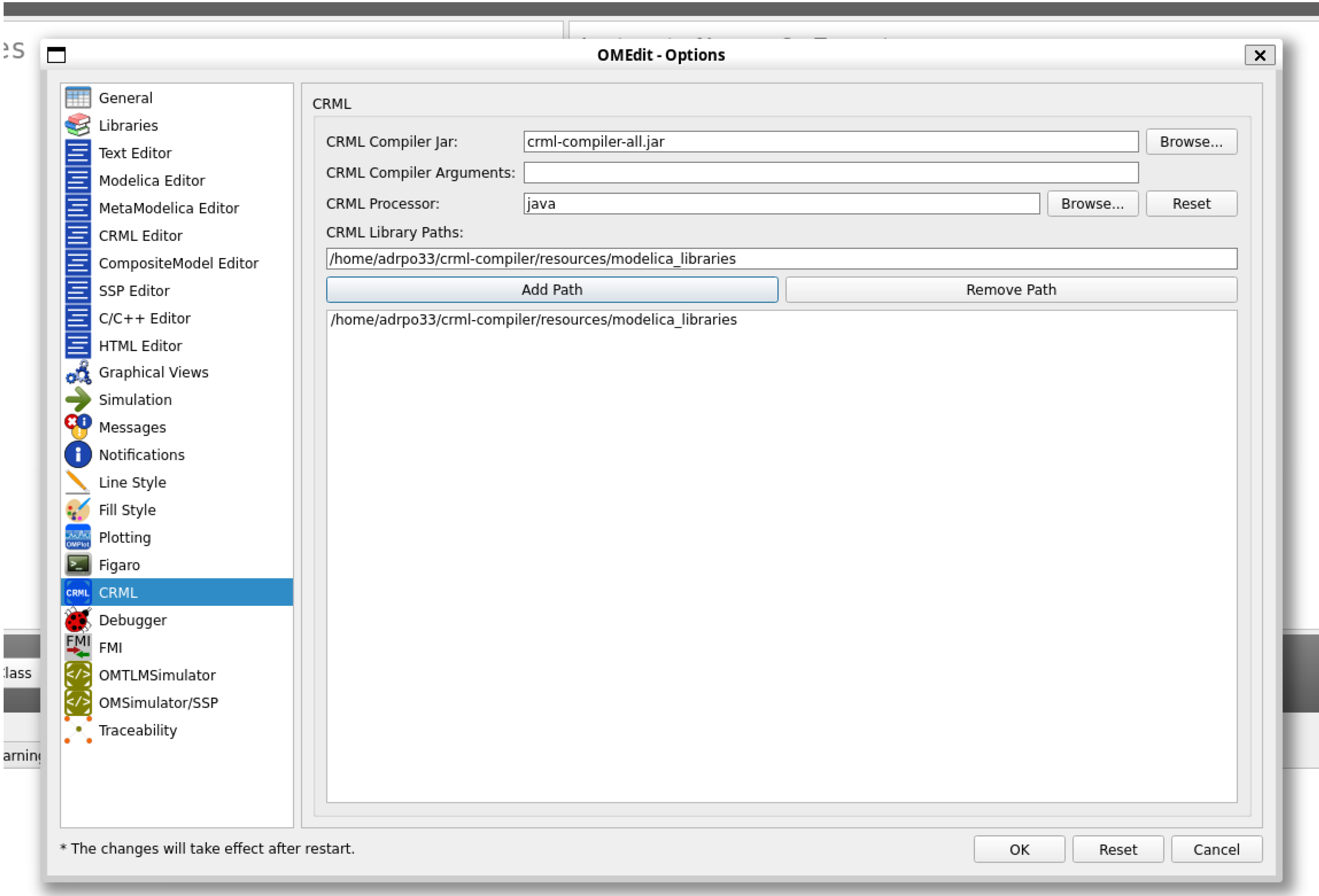
```
model BooleanOperatorsExample1 is {
    // Verify that b1 is equal to b2 when y becomes positive.
```

```
model HelloWorld is {
};
```

* The changes will take effect after restart.

CRML – OMEdit Integration

- Tool Settings



CRML

CRML Compiler Jar: crml-compiler-all.jar

CRML Compiler Arguments:

CRML Processor: java

CRML Library Paths:

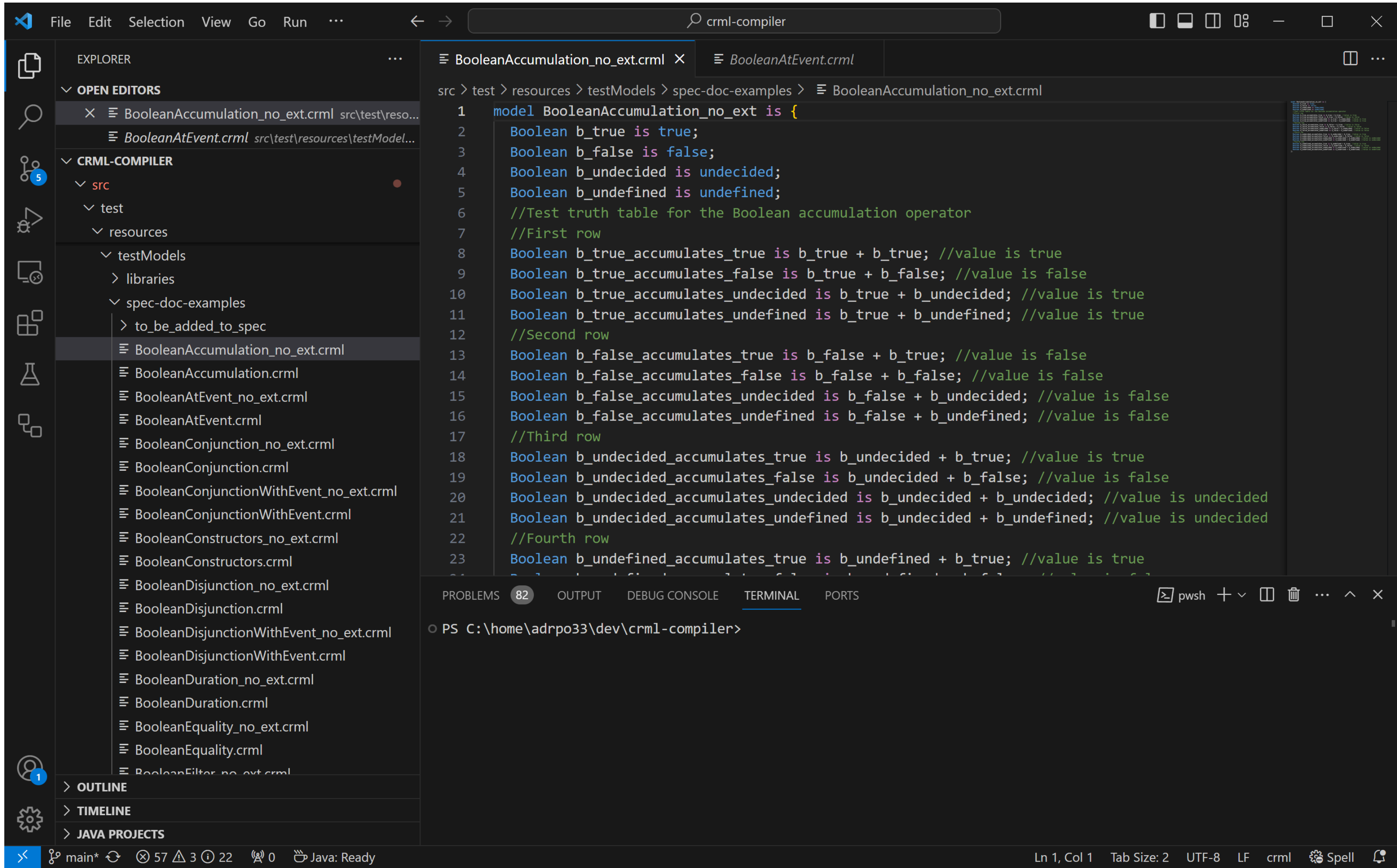
/home/adrpo33/crml-compiler/resources/modelica_libraries

/home/adrpo33/crml-compiler/resources/modelica_libraries

* The changes will take effect after restart.

CRML – VSCode integration

- Basic VSCode extension for CRML
  - https://github.com/lenaRB/crml-vscode
  - syntax highlighting



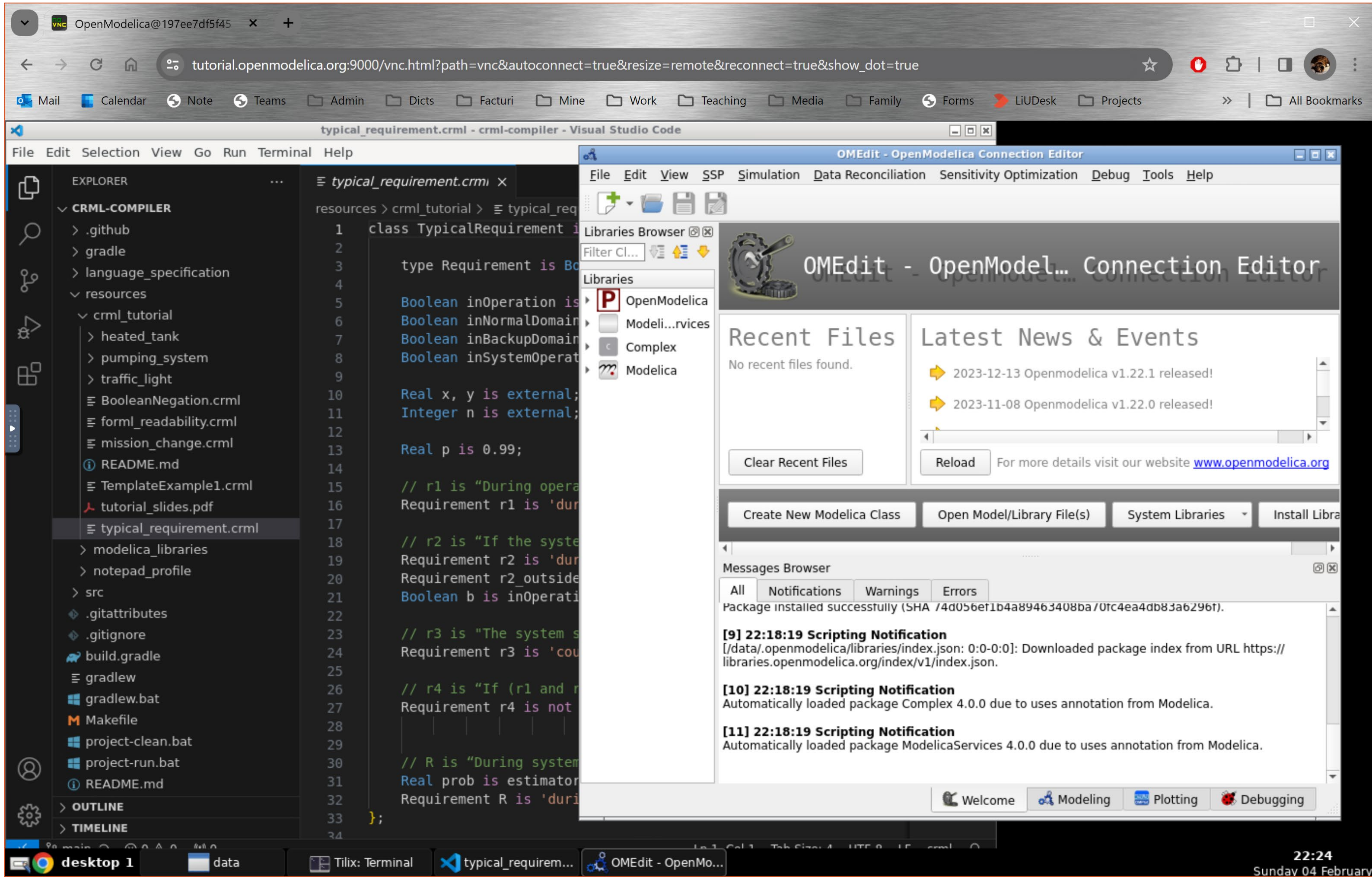
File
Edit
Selection
View
Go
Run
···
crml-compiler
EXPLORER
OPEN EDITORS
BooleanAccumulation_no_ext.crml src\test\reso...
BooleanAtEvent.crml src\test\resources\testModel...
CRML-COMPILER
src
test
resources
testModels
libraries
spec-doc-examples
to_be_added_to_spec
BooleanAccumulation_no_ext.crml
BooleanAccumulation.crml
BooleanAtEvent_no_ext.crml
BooleanAtEvent.crml
BooleanConjunction_no_ext.crml
BooleanConjunction.crml
BooleanConjunctionWithEvent_no_ext.crml
BooleanConjunctionWithEvent.crml
BooleanConstructors_no_ext.crml
BooleanConstructors.crml
BooleanDisjunction_no_ext.crml
BooleanDisjunction.crml
BooleanDisjunctionWithEvent_no_ext.crml
BooleanDisjunctionWithEvent.crml
BooleanDuration_no_ext.crml
BooleanDuration.crml
BooleanEquality_no_ext.crml
BooleanEquality.crml
OUTLINE
TIMELINE
JAVA PROJECTS
BooleanAccumulation_no_ext.crml
BooleanAtEvent.crml
src > test > resources > testModels > spec-doc-examples > BooleanAccumulation_no_ext.crml
PROBLEMS 82
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
pwsh
main*
57 3 22
0
Java: Ready
Ln 1, Col 1
Tab Size: 2
UTF-8
LF
crml
Spell

```
model BooleanAccumulation_no_ext is {
  Boolean b_true is true;
  Boolean b_false is false;
  Boolean b_undecided is undecided;
  Boolean b_undefined is undefined;
  //Test truth table for the Boolean accumulation operator
  //First row
  Boolean b_true_accumulates_true is b_true + b_true; //value is true
  Boolean b_true_accumulates_false is b_true + b_false; //value is false
  Boolean b_true_accumulates_undecided is b_true + b_undecided; //value is true
  Boolean b_true_accumulates_undefined is b_true + b_undefined; //value is true
  //Second row
  Boolean b_false_accumulates_true is b_false + b_true; //value is false
  Boolean b_false_accumulates_false is b_false + b_false; //value is false
  Boolean b_false_accumulates_undecided is b_false + b_undecided; //value is false
  Boolean b_false_accumulates_undefined is b_false + b_undefined; //value is false
  //Third row
  Boolean b_undecided_accumulates_true is b_undecided + b_true; //value is true
  Boolean b_undecided_accumulates_false is b_undecided + b_false; //value is false
  Boolean b_undecided_accumulates_undecided is b_undecided + b_undecided; //value is undecided
  Boolean b_undecided_accumulates_undefined is b_undecided + b_undefined; //value is undecided
  //Fourth row
  Boolean b_undefined_accumulates_true is b_undefined + b_true; //value is true
```

```
PS C:\home\adrpo33\dev\crml-compiler>
```

CRML – Online

- CRML and OpenModelica tutorial available online
  - https://tutorial.opennmodelica.org/
  - No install needed
  - Contact us for user access



```
class TypicalRequirement i

    type Requirement is Bo

    Boolean inOperation is
    Boolean inNormalDomain
    Boolean inBackupDomain
    Boolean inSystemOperat

    Real x, y is external;
    Integer n is external;

    Real p is 0.99;

    // r1 is “During opera
    Requirement r1 is 'dur

    // r2 is “If the syste
    Requirement r2 is 'dur
    Requirement r2_outside
    Boolean b is inOperati

    // r3 is "The system s
    Requirement r3 is 'cou

    // r4 is “If (r1 and r
    Requirement r4 is not


    // R is “During system
    Real prob is estimator
    Requirement R is 'duri
};
```

```
Package installed successfully (SHA 74d056ef1b4a89463408ba70fc4ea4db83a6296f).

[9] 22:18:19 Scripting Notification
[/data/.openmodelica/libraries/index.json: 0:0-0:0]: Downloaded package index from URL https://
libraries.openmodelica.org/index/v1/index.json.

[10] 22:18:19 Scripting Notification
Automatically loaded package Complex 4.0.0 due to uses annotation from Modelica.

[11] 22:18:19 Scripting Notification
Automatically loaded package ModelicaServices 4.0.0 due to uses annotation from Modelica.
```

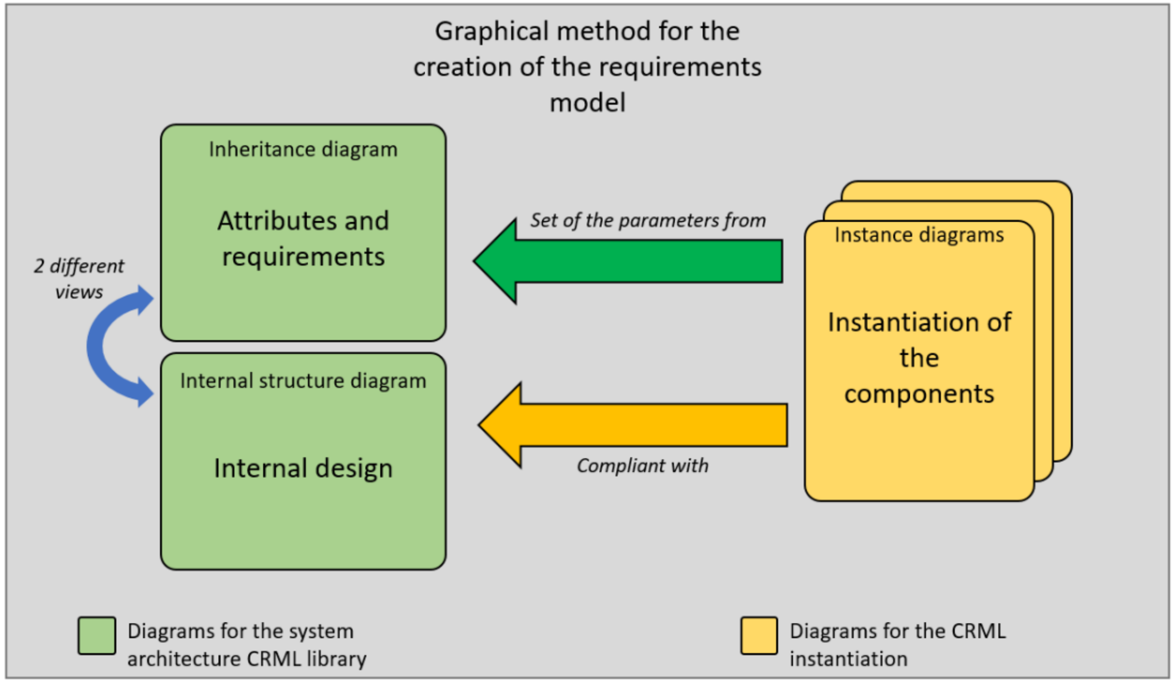
Status (I)

- CRML integration is now part of OpenModelica v1.25
  - Available as nightly-build
  - Will be released as final version soon

- We continue the CRML compiler development
  - To support as much as possible from the CRML specification
    - Work ongoing on support of ETL and FORML libraries, Probabilistic aspects, hierarchical modeling

Status (II)

- Visual Graphical support for CRML
  - Starting point is the KTH Master Thesis from Baptiste Mazurié
  - Support for UML-style (inheritance) class diagrams for CRML classes, attributes and inheritance
  - Support for UML-style internal structure diagrams for CRML class composition and behavior
  - Support for instance diagrams


Graphical method for the creation of the requirements model
Inheritance diagram
Attributes and requirements
Internal structure diagram
Internal design
2 different views
Set of the parameters from
Compliant with
Instance diagrams
Instantiation of the components
🟩 Diagrams for the system architecture CRML library
🟨 Diagrams for the CRML instantiation

Outline

- What is CRML
  - The Common Requirement Modelling Language
- CRML Tooling
  - The CRML Compiler
- CRML Integration
  - OMEdit & VSCode & Online
  - Status
- Future work

Future Work

- Near Future
  - Design and implement the CRML graphical support
  - Present CRML to Modelica Association as a new standard

- Future
  - How to group together several requirements into a project
  - How to handle debugging (CRML <- Modelica <- C code)
  - Evaluate traceability from CRML to simulation results
  - Integration with dashboards to support dynamic requirement monitoring

Thank You!
Questions?

The CRML Project
https://crml-standard.org/

The OpenModelica Project
https://www.OpenModelica.org