OpenBuildingControl

Modelica and related standards for CDL

—

Michael Wetter

February 3, 2021

Lawrence Berkeley National Laboratory

Overview

Goals:

Clarify background so we don't keep getting hang up in discussing wrong statements such as
… Modelica is a suite of tools…
… Modelica is a simulation environment…
… controllers need to run Modelica…
… CDL is a programming language…
… CDL cannot be changed as it requires changing the Modelica Standard…

and prevent making wrong decisions because we misunderstand the technology basis

1) What are Modelica and related standards

2) How does it work

3) What is relation of CDL to Modelica

4) Suggested next steps for committee

BERKELEY LAB

Foundational Standards – Modelica

https://modelica.org/

Open, industry-driven standard for multi-physics modeling.

Used in industry since 2000.

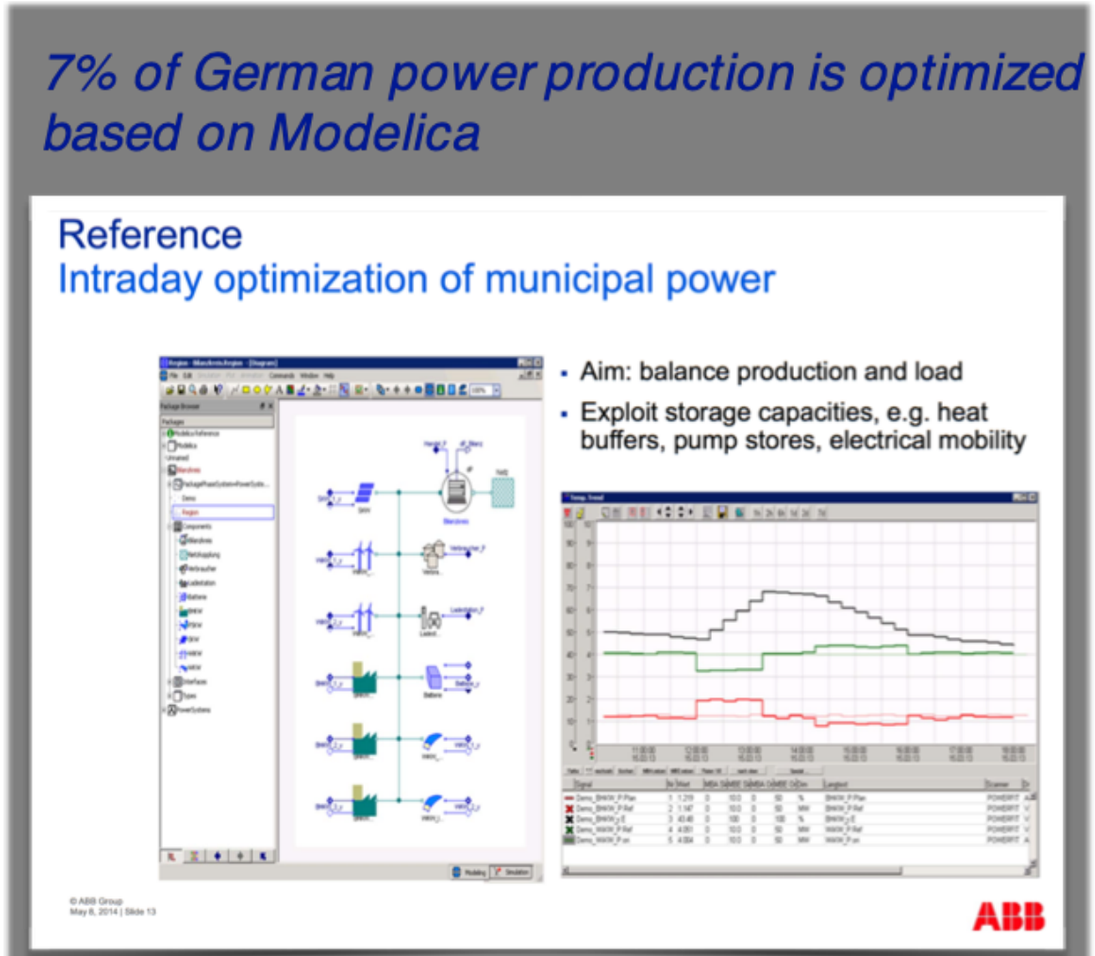
Extensible without having to violate the standard.

Large ecosystem of free and commercial libraries and tools.

Large national and international collaborations on Modelica for building and district energy systems:

- IEA EBC Annex 60:
  42 institutes (2012-2017)
- IBPSA Project 1:
  >80 FTE, 29 institutes + 51 individual participants (2017-2022)
- Spawn of EnergyPlus:
  DOE funded with lab/industry to add Modelica-based control capabilities to EnergyPlus
- OpenBuildingControl:
  Modelica-based building control sequence implementation.
- BOPTEST:
  Modelica- & FMI-based testing framework for building control.
- URBANopt District Energy Systems:
  2 DOE funded projects for Modelica-based tools for district energy systems and waste heat utilization
- Co-design:
  2 DOE funded projects for Modelica-based co-design of controls, HVAC & electrical grid

7% of German power production is optimized based on Modelica



Source: http://new.abb.com/power-generation/power-plant-optimization

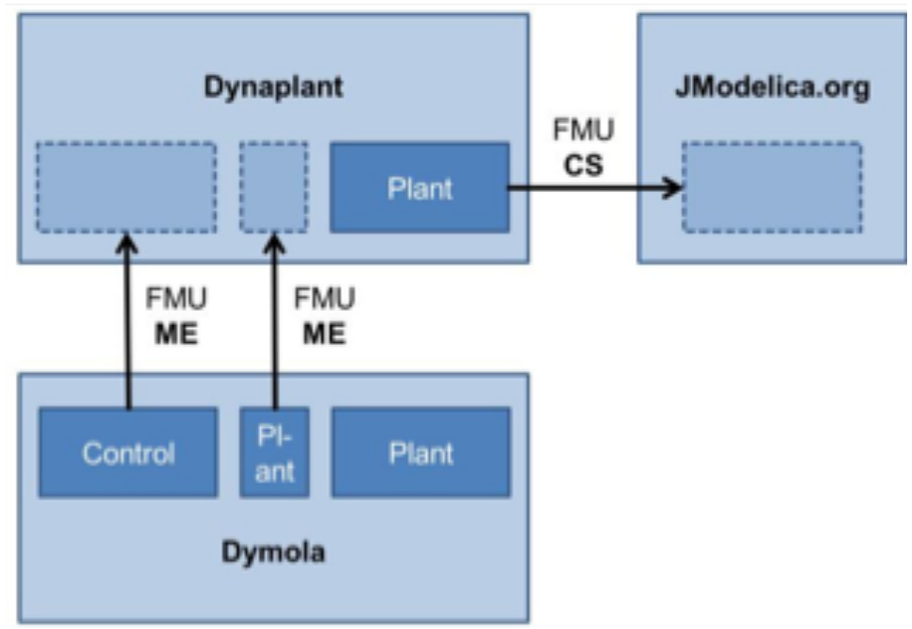
Related standards

https://www.fmi-standard.org/
API standard to exchange simulators or models

ITEA project, 28 partners, 178 person years, 26 Mill. € budget, July 2008 - June 2011.

First version published in 2010.

Supported by >100 tools.



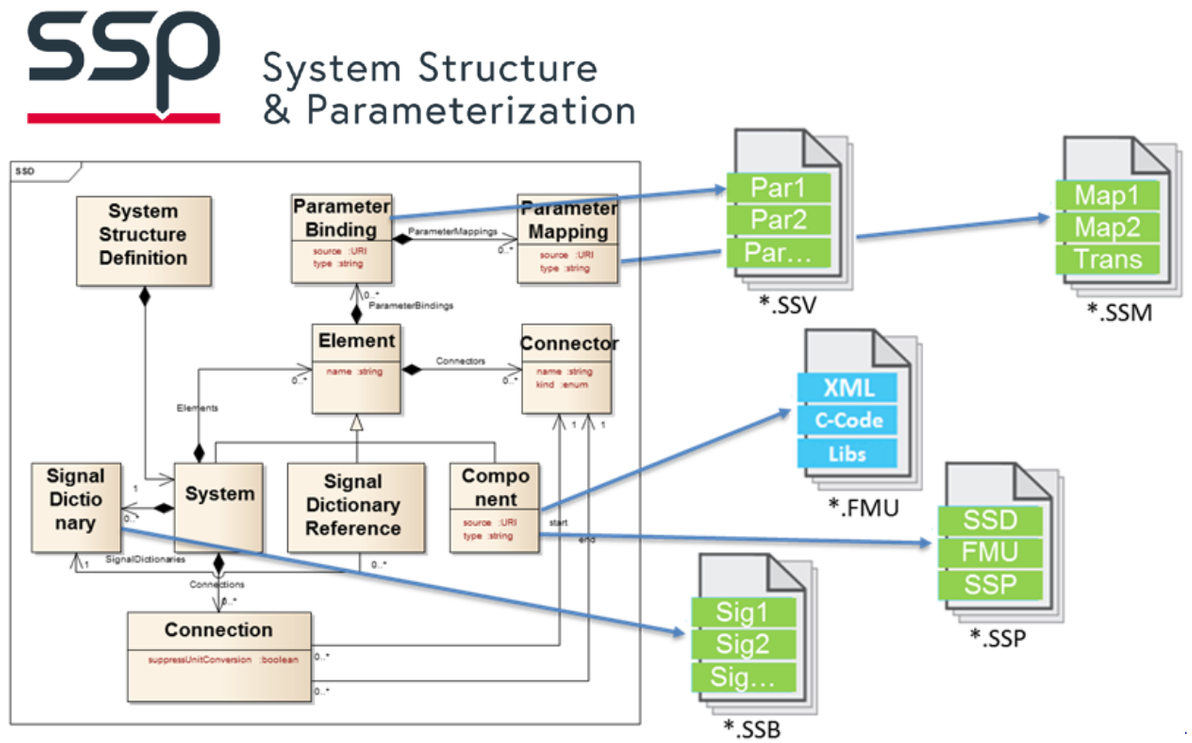
Powerplant simulation with Modelica (Dymola) coupled to in-house simulator (Dynaplant). Source: Siemens, doi:10.3384/ecp1511817

https://ssp-standard.org/

System Structure and Parameterization (SSP)

Tool independent standard to define complete systems consisting of one or more FMUs, including its parameterization that can be transferred between tools.

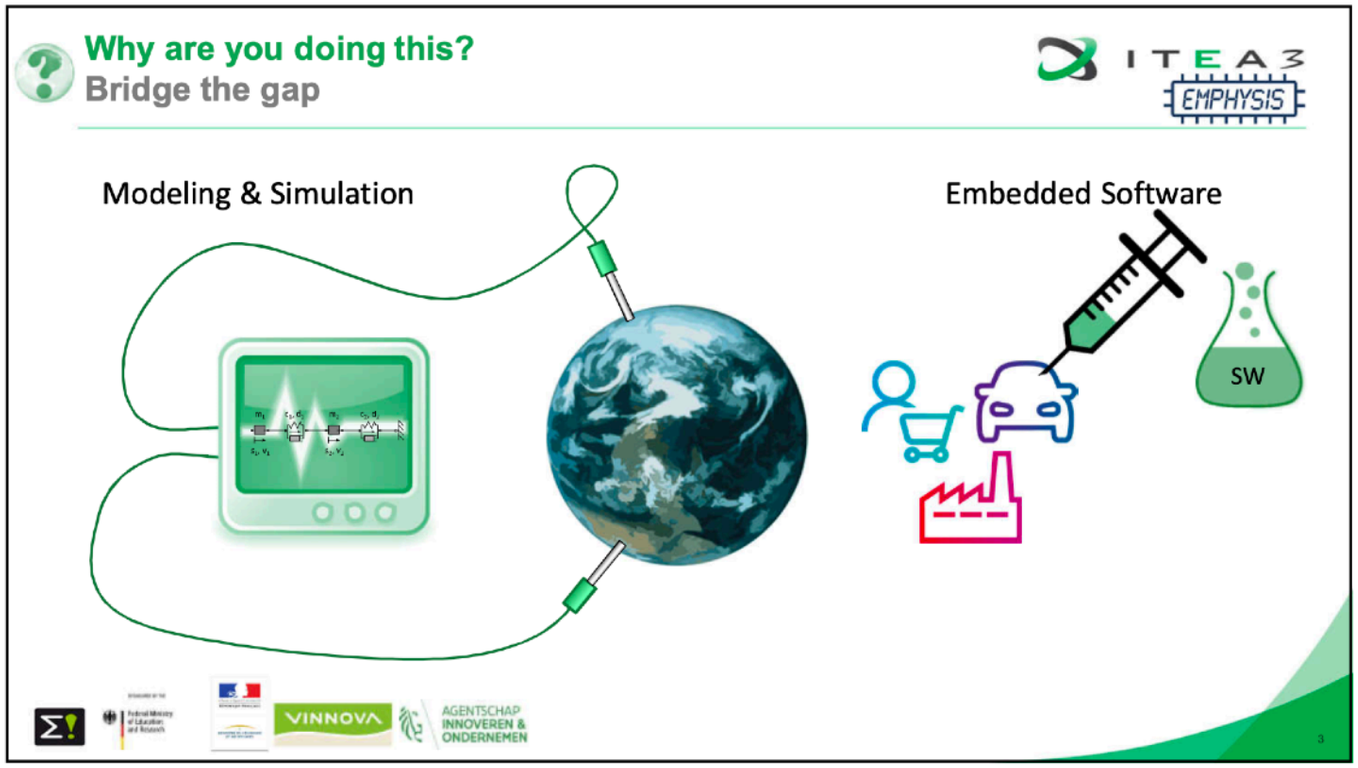
First version published in 2019.

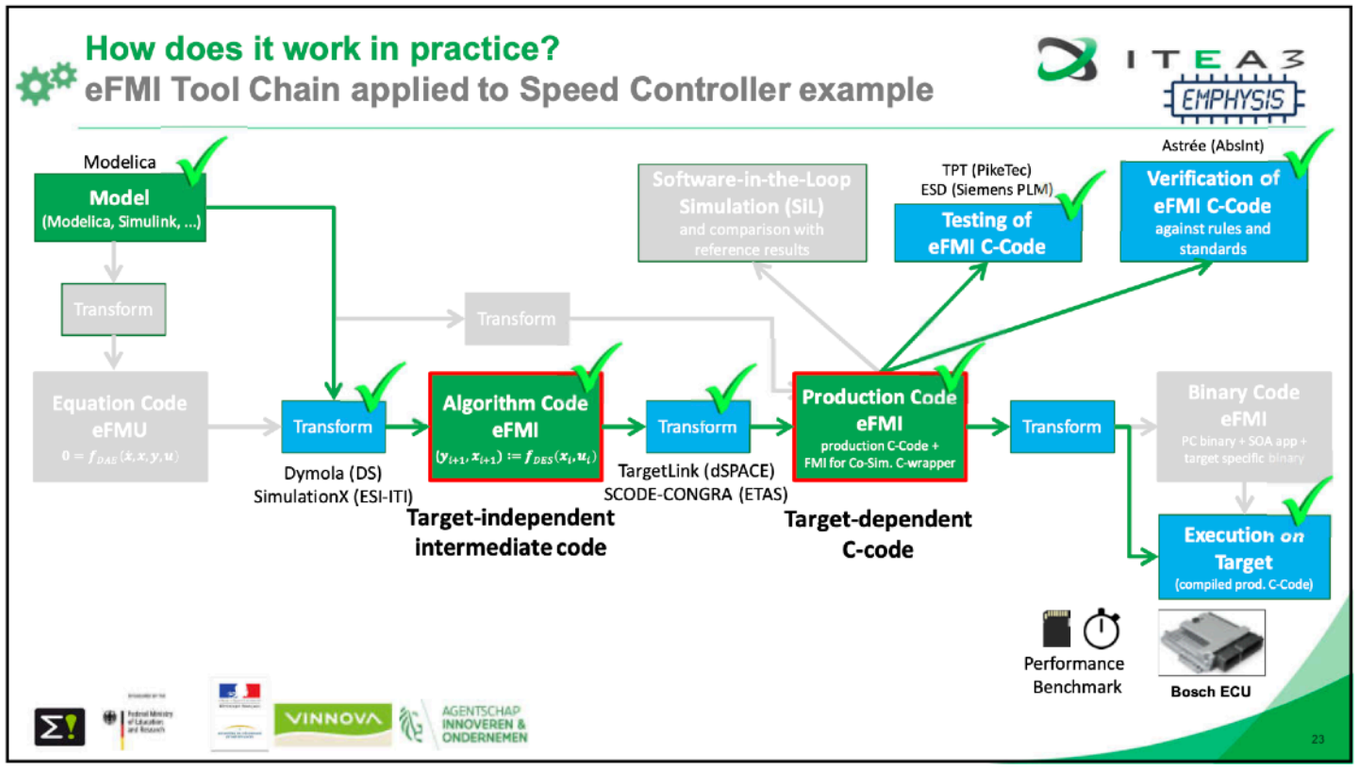


Related standards

eFMI
Expected early 2021.

https://modelica.github.io/Symposium2019/slides/jubilee-symposium-2019-slides-lenord.pdf





BERKELEY LAB

Why are these standards important?

Robust, rigorous, well-defined basis, no need to reinvent the wheel.

Provides stable basis for industry investment.

Avoids vendor lock-in and single dependency on a provider.

Ecosystem of developers and users provides tools for

- sequence authoring
- simulation for
  - control sequence development and testing and
  - performance assessment with energy model in the loop
- documentation
- code generation (for various platforms)
- translation to various other formats

What is special about Modelica?
Separation of concerns, as is used in mainstream computer applications

Web

Rendered page

html & json: data declaration

css: layout

Computers

User interface

Operating system

Hardware drivers

Modelica

Icons that encapsulate models & expose their I/O.

Equations & connect statements (that generate equations)

Solvers

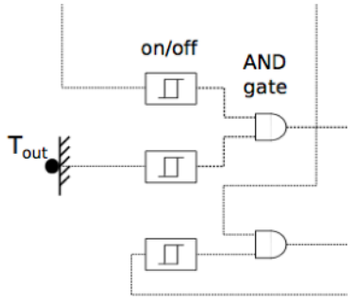
Code generators

Simulation programs

Input/output routines + solvers + data I/O all intertwined…

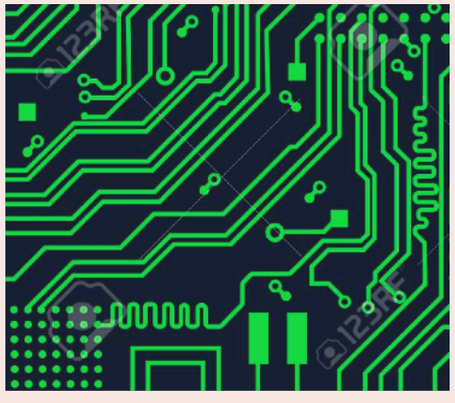
BERKELEY LAB

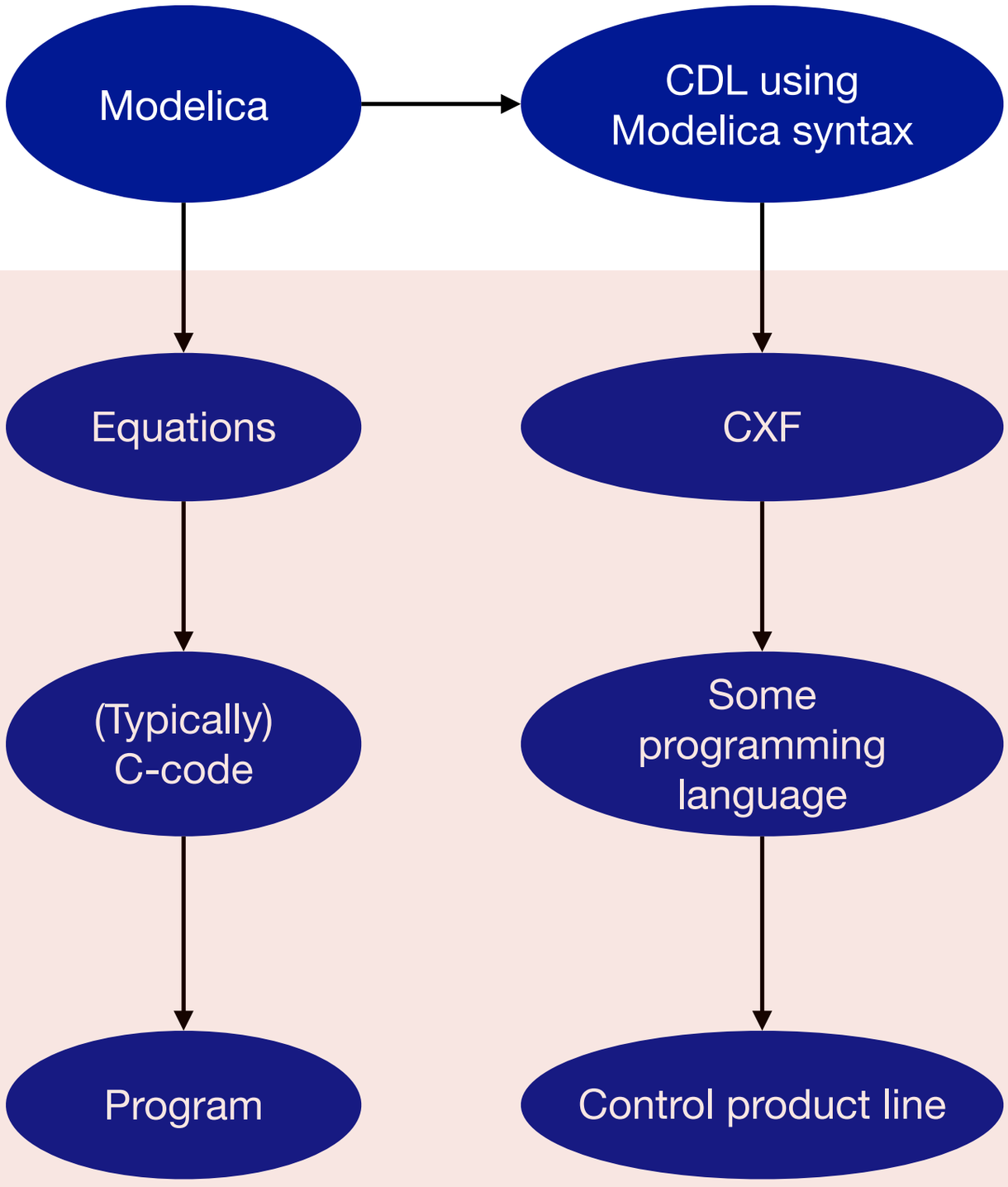
Users interact with declarative, human-readable representation, and not with lower-level representations




```
der(bouA.dynBal.mXi[1]) = bouA.
0 = - doo.m1_flow + doo.m2_flow
der(bouA.dynBal.U) = bouA.dynBa
der(bouA.dynBal.U) = bouA.dynBa
der(bouA.dynBal.mXi[1]) = bouA.
bouA.dynBal.medium.MM = 1 / (bo
bouA.dynBal.medium.dT = bouA.dy
doo.port_a2.h_outflow = bouA.dy
```

```
*dQConSen_flow = (QConSenPer_flow
/* Get next event time, unless FM
if (bui->mode == initializationMo
  if (bui->logLevel >= MEDIUM)
    SpawnFormatMessage("%.3f %s:
       bui->time, zone->modelicaN
  *tNext = bui->time; /* Return s
}
else{
  if (bui->logLevel >= TIMESTEP)
```





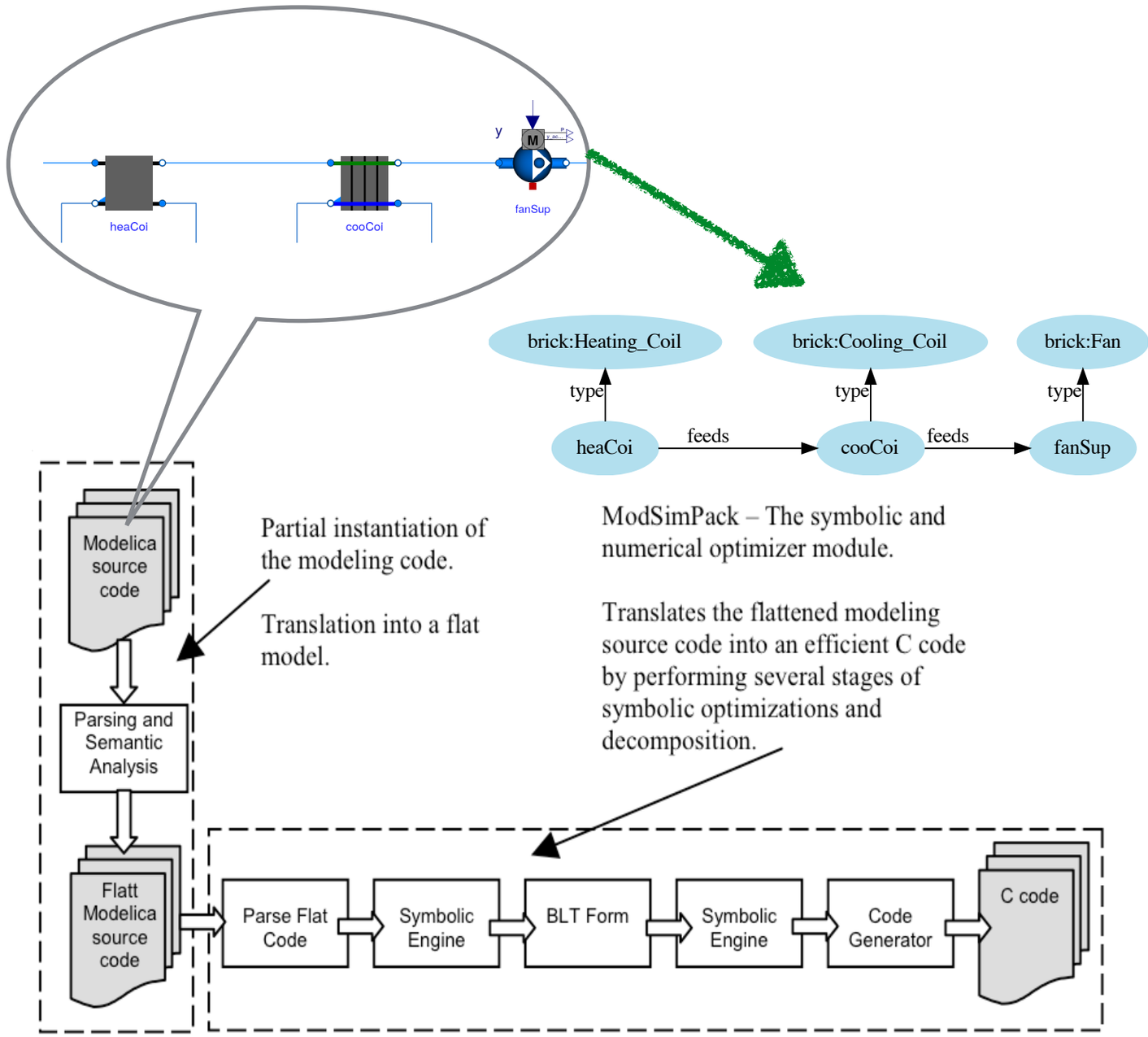
Modelica modeling and simulation tools

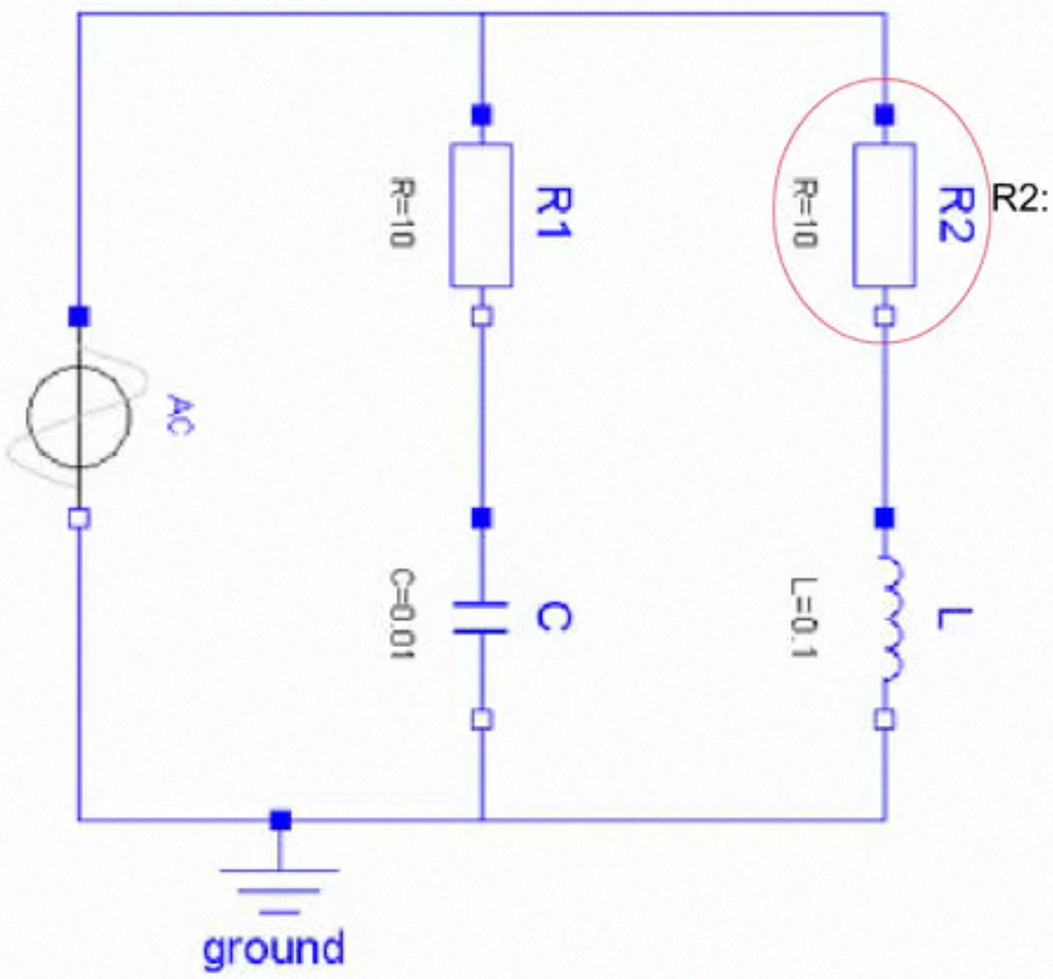
Commercial

1. Altair: solidThinking Activate
2. Ansys: Twin Builder
3. Dassault Systèmes: Dymola
4. ESI ITI: Simulation X
5. Maplesoft: MapleSim
6. Modelon: OPTIMICA
7. Siemens: Simcenter Amesim
8. Suzhou Tongyuan: Builder
9. Wolfram: SystemModeler

Free

10. OpenModelica
11. JModelica





DAE:

R1: R1.v = AC.Vp - R1.Vn

R1.R * R1.i = R1.v

R2: R2.v = AC.Vp - L.Vp

R2.R * L.i = R2.v

Movie courtesy of Modelon

CDL can but does not require to use such advanced tools

CDL is a subset of Modelica designed to not require use of these advanced methods,

but allows to use them to ensure that simulations are fast and correct (using free & open-source tools).

What is Modelica not?

Not a programming language.

- -> its behavior is expressed mathematically, how to execute it is intentionally not specified.

Not a software tool implementation.

- It is declarative.
- How to edit, debug, simulate, show documentation etc. is up to tools that implement it.

BERKELEY LAB

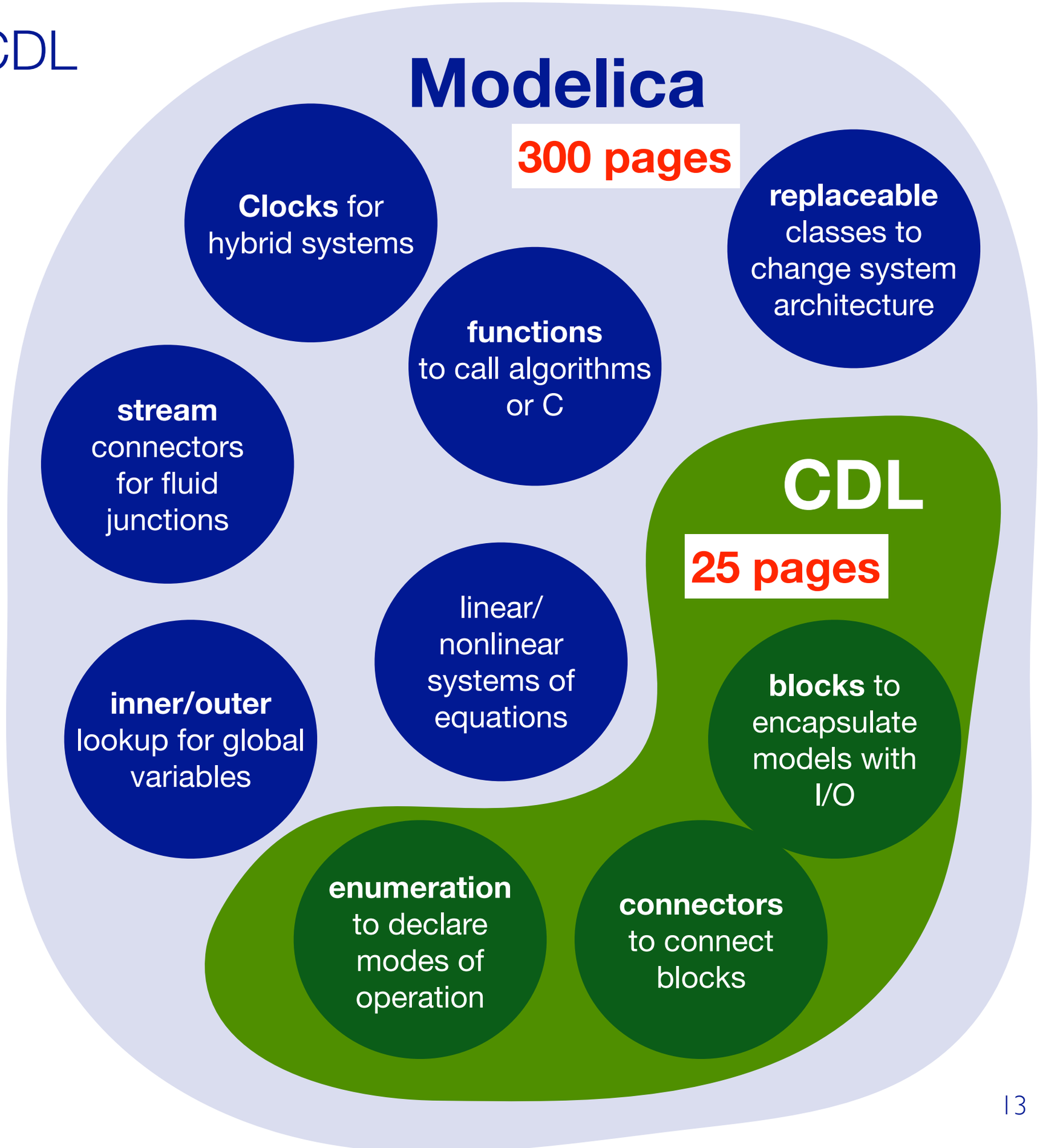
How we specified CDL

Conform to the Modelica Standard 3.3, ***but*** remove everything that is not needed to practically declare control logic and their English language documentation.

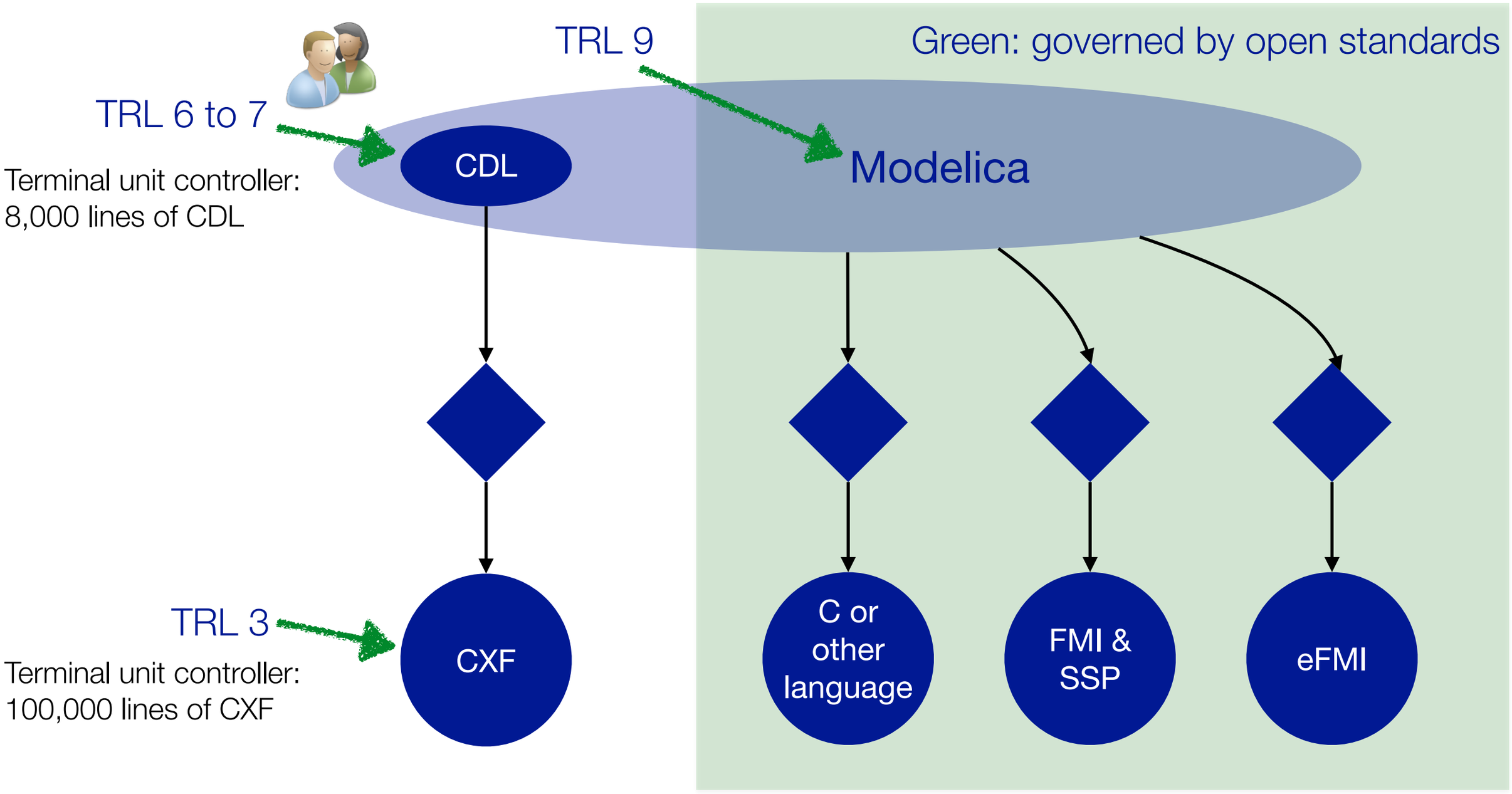
Keep it simple & easy to parse.

… and allow reuse of technology from the Modelica ecosystem.

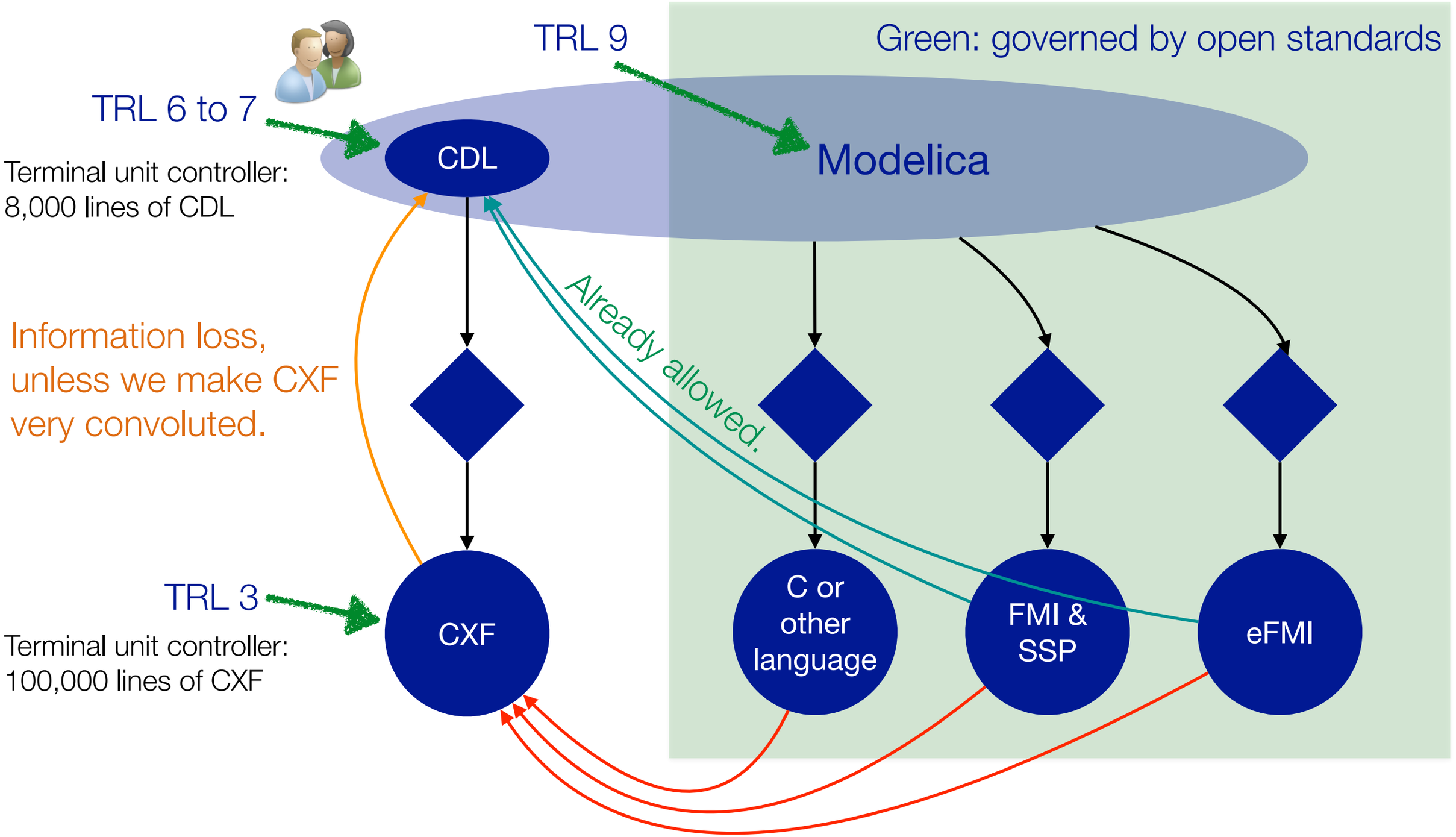
Reviewed by advisory panel, through peer-review process, and used by various project partners from industry.



CXF is at lowest TRL level,
and not related to or integrated with open standards.



CXF is at lowest TRL level, and not related to or integrated with open standards.



None of these red paths makes sense.
They would be very convoluted.

Suggested next steps

1. Be explicit about use cases.
2. Write down requirements.
3. Get familiar with existing work:
    - Read the 25 pages of the current CDL specification.
    - Understand related open and extensible standards.
4. Understand how we test for compliance with the standard.
5. Discuss what should be in Standard 223P.

BERKELEY LAB