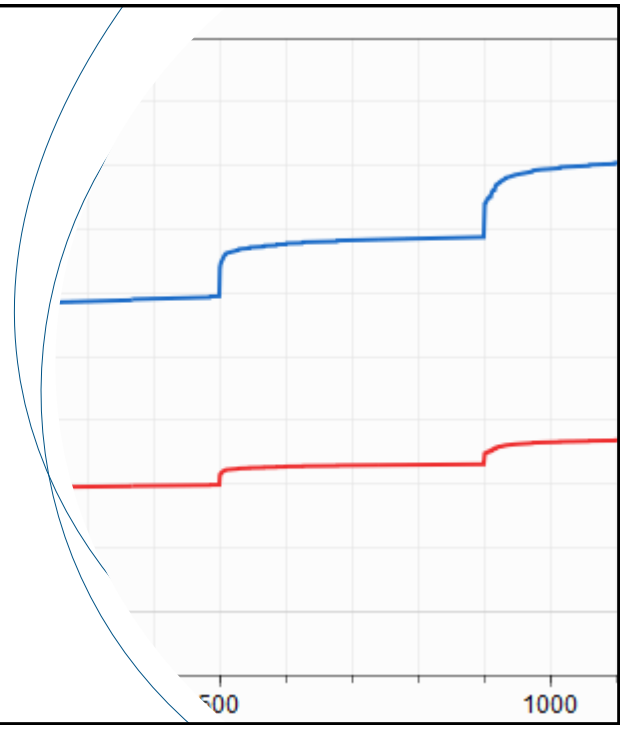
DASSAULT SYSTEMES

DYMOLA 2025X HIGHLIGHTS

29 November 2024

3DEXPERIENCE



EXECUTIVE SUMMARY

Model development

- Variable-length parameter arrays
- Improved parameter management
- Better Git support

Simulation

- Faster simulation of Modelica functions
- New FMI co-simulation technology
- Dymola Modelica Compiler

Libraries

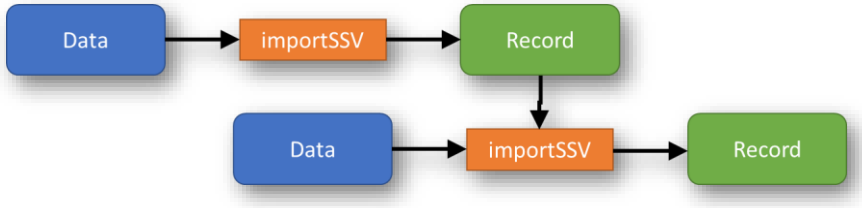
- ThermalSystems replaced by TIL Suite

MODEL DEVELOPMENT

DASSAULT SYSTEMES

PARAMETER MANAGEMENT IN DYMOLA

- Variable-length parameter arrays
  - No need to use external table objects
  - Initialize parameters (including arrays) at simulation initialization, not translation
  - Also supported in FMUs
- Parameter records
  - Import parameter set to create the corresponding parameter record
  - Modify one parameter set (record) with another
  - SSV and CSV supported



DASSAULT SYSTEMES

VARIABLE-LENGTH PARAMETER ARRAY

- Possible to declare parameter arrays which size is not fixed at translation time
  - This a Dymola extension controlled by an annotation
- No need to use external object (table) – directly supported in “native” Modelica
  - Can use interpolation in e.g. Modelica vectors
- Variable-length arrays are not stored in the result file

```
model Test
  parameter Real p[:] = InitializeParameters("parameterfile.txt“)
     annotation(__Dymola_UnknownArray=true);

  extends Modelica.Mechanics.Rotational.Examples.CoupledClutches(
    J1(J=p[1]),
    J2(J=p[2]),
    J3(J=p[3]),
    J4(J=p[4]));
end Test;
```


VARIABLE-LENGTH PARAMETER ARRAY

- Can read external file to initialize at runtime
  - Trivial example reading a text file

```
function InitializeParameters "Reads initial values for parameter set"
  input String filename "File to read parameters from";
  output Real p[:] "Parameter set from file";

protected
  Integer n = Modelica.Utilities.Streams.countLines(filename);
  String data[:] = Modelica.Utilities.Streams.readFile(filename);

algorithm
  p := fill(0.0, n);  // Initialization needed to get the right size

  for i in 1:n loop
    p[i] := Modelica.Utilities.Strings.scanReal(data[i]);
  end for;
end InitializeParameters;
```

CREATE PARAMETER RECORD FROM DATA

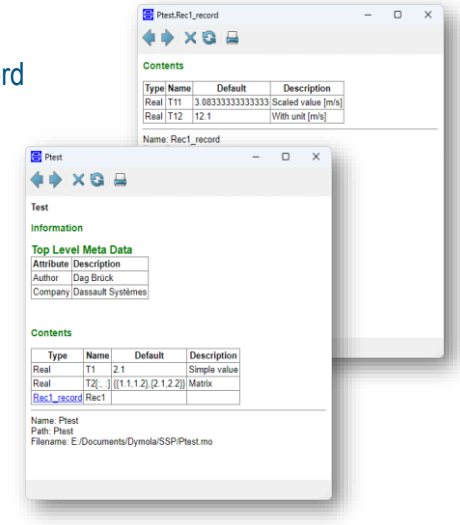
- Dymola can automatically create a Modelica parameter record
  - Using a data file as template (and for default values)
  - Handles name, description, unit conversion, nested records for names with dot
  - Man modify existing parameter record
  - Supports SSV and CSV data formats

```
importSSV("Partest.ssv");
```

```
record Partest "Test"
  parameter Real T1=2.1 "Simple value";
  parameter Real T2[:,:]={{1.1,1.2},{2.1,2.2}} "Matrix";
  record Rec1_record
    parameter Real T11(
      unit="m/s",
      displayUnit="km/h") = 3.08333333333333 "Scaled value";
    parameter Real T12(unit="m/s") = 12.1 "With unit";
  end Rec1_record;
  parameter Rec1_record Rec1;
end Partest;
```

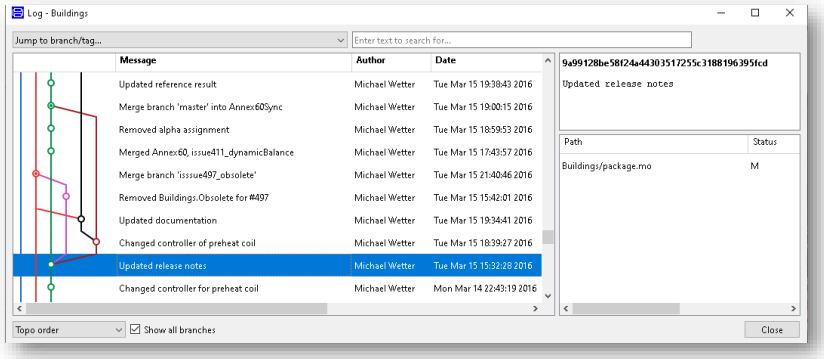

PARAMETER RECORDS

- Best practice is to manage all global parameters as a record
  - Instead of calling file reading functions all over the model
  - More practical when variable-length arrays are supported
- SSP provides packaging of
  - Simulation model
  - Parameter sets
  - Documentation
- Improved documentation of records in Dymola



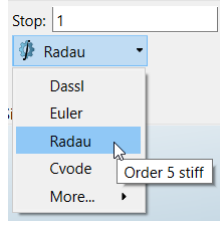
VERSION MANAGEMENT WITH GIT

- New dialog to display the version log
  - Graphical display of branches and merges
  - Commit message, files changed by commit



SMALLER IMPROVEMENTS

- Improved unit checking
  - Expressions, built-in functions
  - Considers value of evaluable parameters and only active equations
  - Unit checking on array elements
- Multi-lingual support for libraries
  - According to the MSL specification
- Text string expansion of description in the graphical layers (previously only name)
  - `%classdescription`
  - `%componentdescription`
- Simulation Analysis is now called Performance
- Simplified and grouped ribbon menu for integration algorithms

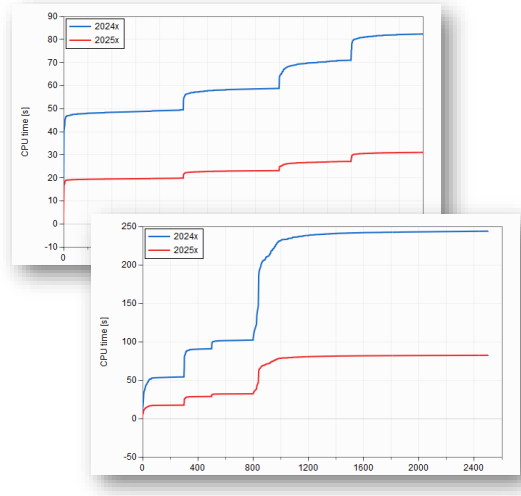


SIMULATION

SIMULATION SPEED

- Improved code generation for Modelica functions
  - Improved array handling in functions
  - Constant and parameter optimizations
  - Typically good effect in fluid systems with many function calls for media calculations
- Further optimizations by enabling

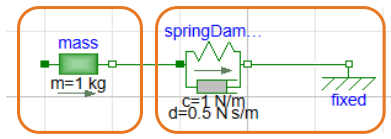
  `Advanced.Beta.Translation.FunctionsUseConstantsInsteadOfExpressions`
  - May require setting extra start values in the model



Examples from ThermoFluidStreams

FMI CO-SIMULATION TECHNOLOGY

- New co-simulation technology aimed to
  - Improve performance of ”heavy” FMUs with variable-step solvers
  - Reduces number of f-evaluations and Jacobian evaluations
  - Collaboration with partner TLK-Thermo
- Smoothing of continuous-time Real inputs
  - Linear interpolation during the next doStep
  - Integrator can continue without costly reset
    → larger step-size, fewer evaluations
  - Predictor compensation → better error estimates
- Caveats
  - Inputs are effectively delayed
  - Not a universal cure for all co-simulation problems



Input smoothing	# f-evals	# Jacobian evals
Default	5471	499
Enabled	1851	14

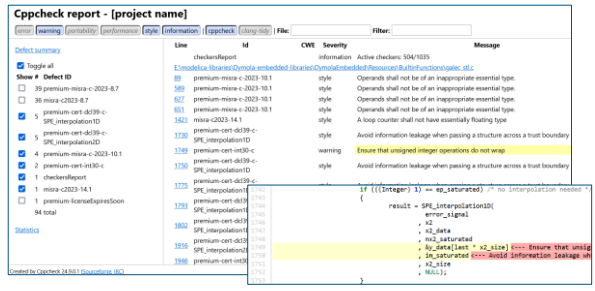
SYSTEM STRUCTURE AND PARAMETERIZATION

- SSP 2.0 supported
  - Support for Modelica components in SSP
  - Better FMI 3.0 conformance (Float32 etc.)
  - Local variable can be defined as connector with `kind="local"`
  - Inner and outer connector positions (cf. diagram and icon)
- Initial meta-data support according to SSP Traceability
- Improved SSP conformance, cross-tool compatibility
- Black-box import of FMUs supported by `importSSP`
  - `includeAllVariables=false`

FMI FOR EMBEDDED SYSTEMS

efmi Functional Mock-up Interface for embedded systems

- Algorithm Code
  - Support for untypical, error-case start values in manifests and initialization (NaN, ±∞)
  - Support for many more GALEC builtin functions (3D interpolation, all integer division and remainder variants, …)
- Behavioral Model
  - Derived experiment-packages now support boolean and integer GALEC block-interface in- and outputs
- Production code checks with Cppcheck and clang-tidy
  - Backed by Software Production Engineering on **3D**EXPERIENCE
  - MISRA C:2012, MISRA C:2023, SEI CERT C
  - Strict preconfigurations, with very few exceptions and altera rules for FPGA & CUDA programming)
  - HTML log with syntax highlighting for Cppcheck results
  - Profiles for open source and premium Cppcheck



CATIA

DASSAULT SYSTEMES

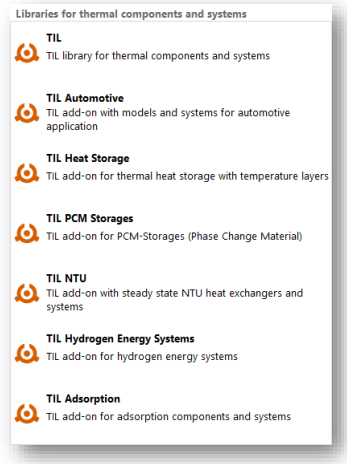
MODELICA LIBRARIES

CATIA

DASSAULT SYSTEMES

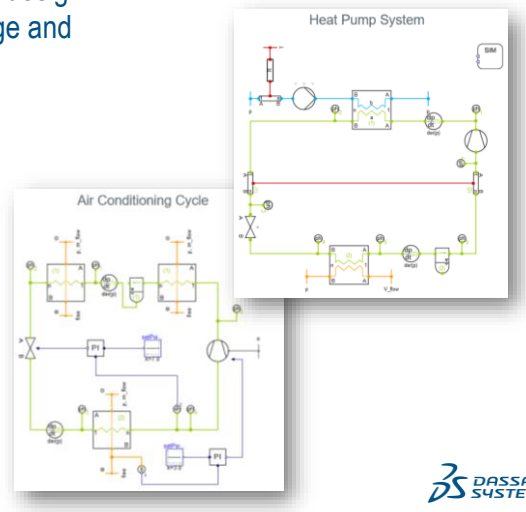
TIL-SUITE LIBRARIES

- ThermalSystems library is replaced by TIL-Suite
  - More complete range of libraries, extended functionality
- Divided into four library products at Dassault Systèmes
  - TIL Base Library (FNY-x)
  - TIL Mobile Air Conditioning Library (HMY-x)
  - TIL Hydrogen Library (HNY-x)
  - TIL Thermal Storages Library (TTY-x)
- Free upgrade for existing customers of ThermalSystems
  - ThermalSystems → TIL Mobile Air Conditioning Library
  - Contact your sales channel (not automatic)



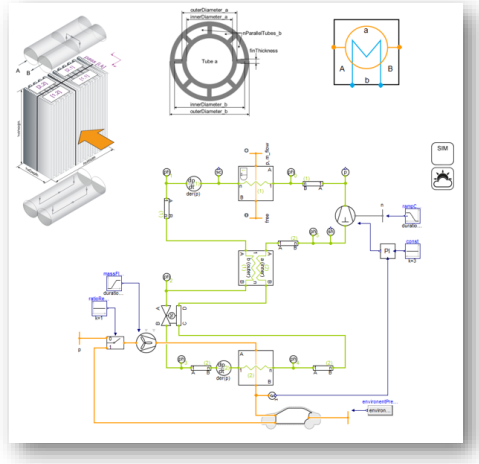
TIL BASE LIBRARY

- Intended for the stationary and transient simulation, design and optimization of individual components up to large and complex systems
  - Refrigeration cycles, including refrigeration mixtures
  - Heat pump systems e.g. with ejectors
  - Hydraulic networks
  - Rankine cycles
  - Heating, ventilation and air-conditioning systems
- Includes the TIL Media library
- Foundation for all the other TIL-Suite libraries



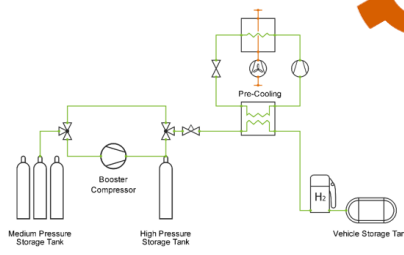
TIL MOBILE AIR CONDITIONING LIBRARY

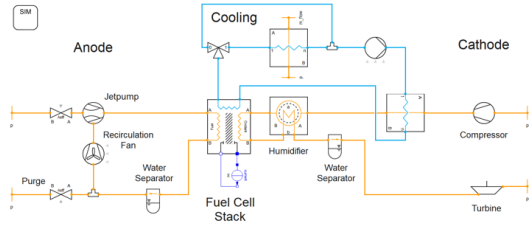
- Focuses on mobile air conditioning systems with models for
  - Car, coach and train cabins
  - Detailed MPET heat exchanger (configurable flow patterns)
  - Internal heat exchanger
  - Common example systems for mobile AC cycles with different refrigerants
  - Steady-state heat exchangers using a “Number of Transfer Units” (NTU) method
- Includes all of TIL Base



TIL HYDROGEN LIBRARY

- Simulate systems of the entire hydrogen value chain
  - H_2 production and utilization
  - H_2 storage and distribution
  - Applications – fuel cell systems
- Simulation and analysis of adsorption processes
  - Drying applications, gas separation or direct air capture
  - Models are based on the physical principle of adsorption
  - Extendable with material data for different adsorbents





TIL THERMAL STORAGES LIBRARY

- Hot water storage tank
  - Stratified temperature layers
  - Used for example in residential heat pump systems
  - Optional internal and mantle heat exchangers
  - Models for buoyancy, walls and insulation
- Phase Change Material (PCM) Storages library
  - Different types of geometries and fluid combinations for thermal PCM storages / heat exchangers
  - Properties of solid-liquid equilibrium (SLE) media, such as cold ice storage

TIL-SUITE LIBRARY / PRODUCT MATRIX

TIL Suite	TIL Base Library (FNY-x)	TIL Mobile Air Conditioning Library (HMY-x)	TIL Hydrogen Library (HNY-x)	TIL Thermal Storages Library (TTY-x)
TIL	X	X	X	X
TIL Automotive		X		
TIL Heat Storage				X
TIL PCM Storage				X
TIL NTU		X		
TIL Hydrogen Energy Systems			X	
TIL Adsorption			X	
TIL Cabin		X		
TIL Media	X	X	X	X

DYMOLA MODELICA COMPILER

DASSAULT SYSTEMES

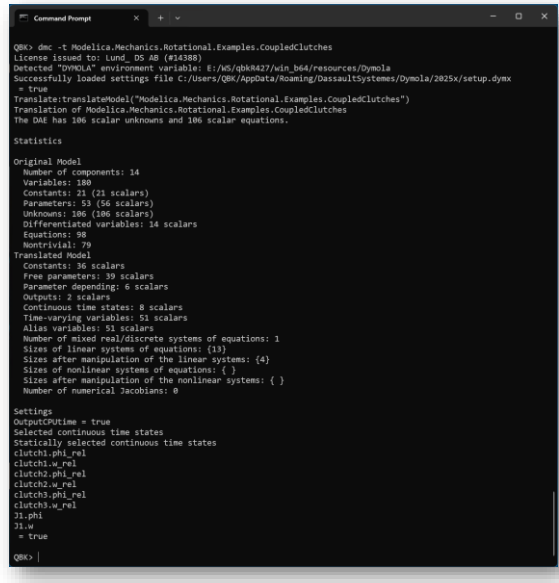
DYMOLA MODELICA COMPILER

- Command-line Dymola
  - Dymola compiler without any graphical user interface
  - Supports non-gui toolchains
  - For people who like black&white movies
- Limited in what it can do
  - Translates and simulates models
  - Runs scripts
  - Supports Python/Java/JavaScript interface via network port
- Uses normal Dymola license

DASSAULT SYSTEMES

DYMOLA MODELICA COMPILER

Command	Meaning
-h	Print help
-o <file-name>	Open named Modelica file
-t <model-path>	Translate the named model
-x <command>	Execute the command
-r <file-name>	Run a script file
-p <port>	Start the HTTP server
-nosettings	Do not read settings file



IF WE™