

Dymola Referential

Dag Brück, 11 October 2018

Contents

Introduction.....	1
Dymola Overview	1
Model Based System Engineering	3
Modelica	6
Benefits of Equation-Based Modeling.....	6
Templates and Reuse	8
Model Design Tools.....	9
Code and Model Export.....	10
Interfacing Other Software.....	11
Academic Packages.....	11
User Roles.....	11
Industry Solutions: Model Libraries.....	13
Functional Mockup Interface	16
References.....	17

Introduction

This document is an attempt to summarize benefits and technical overview information for Dymola, Modelica and FMI. The parts can be read independently, based on interest and need. Additional resources are available online at www.dymola.com. Feedback is always welcome.

Increasing product complexity

Products are becoming more complex and the demands for better performance and reduced cost are ever increasing. Much of the complexity arises from a higher degree of interaction between sub-systems. Combined with the demand to reduce time-to-market, the need to increase development productivity is higher than ever.

With Model Based System Engineering, system architectures can be quickly assessed based on behavior and performance simulation. Informed decisions are made earlier in the development process, long before detailed design or physical prototype manufacturing.

Fighting complexity with Dymola

This document describes the benefits of Dymola and the underlying technologies, such as, Modelica and FMI.

After an overview of all Dymola features, we take a step back and try to place behavior modeling and Dymola in the general framework of Model Based System Engineering. Then follows a few chapters focused on the Modelica language and the benefits that follow, and which are effectively capitalized in Dymola. We conclude that part with a look at certain tools and export capabilities in Dymola.

Then follows a chapter we try to look at the benefits from a user role perspective, rather than a tool/technology perspective. Model libraries are key here because they encapsulate application domain knowledge. Finally, we touch upon the Functional Mockup Interface that provides a standardized interface for code modules.

Dymola Overview

Dymola is a Modelica compliant solution that efficiently models and simulates multi-physic dynamic systems. Dymola rapidly solves complex multi-disciplinary systems modeling problems that can contain a combination of mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented characteristics and components.

Dymola Benefits

- Powerful multi-disciplinary systems engineering through compatible model libraries for a large number of engineering domains.
- High-fidelity modeling of complex integrated systems.
- Intuitive modeling.
- Open model libraries enabling users to easily build their own.

Multi-discipline modeling and simulation

The unique cross-discipline support provided by Dymola enables users to define and simulate models that comprise physical components from many engineering domains.

The fundamental system components are described by ordinary differential and algebraic equations resulting in reusable models of complete systems that are more easily maintained than traditional block diagram based modeling approaches.

Intuitive modeling

Dymola's graphical editor and the multi-discipline engineering libraries make modeling fast and easy. The libraries include elements corresponding to physical devices which are simply dragged-and-dropped to build the system model.

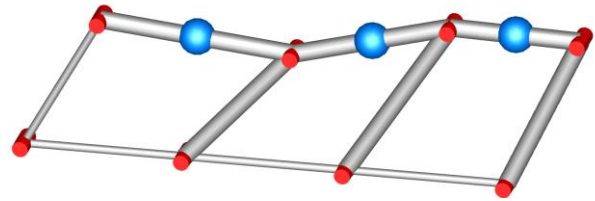
Interactions between the components are conveniently described by graphical connections that model the physical coupling of the components. This means that models are intuitively organized the same way as the physical system is composed.

Outstanding performance

Dymola has unique and outstanding performance for solving systems of differential algebraic equations.

The key to this high performance and robustness is symbolic manipulation that optimizes and

solves the systems of equations that define the system.

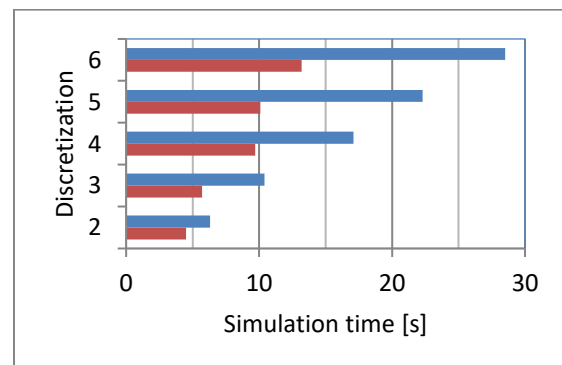


[Mechanism with three planar kinematic loops and one degree of freedom.]

For example, Dymola will optimize the 3D multi-body mechanism above utilizing constraints given by the joints and evaluated parameters. The non-linear equation systems for the coupled planar loops will be solved analytically during the translation. The generated simulation code has only two continuous time states.

These techniques together with special numerical solvers enable real-time Hardware-In-the-Loop simulation of automotive drivelines and full vehicle dynamics models with flexible elements.

For certain kinds of models large performance gains can be achieved by using multiple processor cores. Dymola will automatically analyze the equations and partition the code.



[Speed improvement of using four processor cores for simulating the evaporator of an air conditioning system, with increasing level of model detail.]

Open and flexible

The Modelica based environment is completely open in contrast to many modeling tools that have a fixed set of component models and proprietary methods for introducing new components. Users of Dymola can easily create models that match their own and unique needs. This can be done either from scratch or by using existing components as templates.

Functional Mockup Interface (FMI)

FMI is an open, general and vendor independent interface standard that provides advanced runtime tool interoperability that enables accurate system model compositions to be created. Dymola fully supports FMI, enabling plant models and ECU interactions to be validated using multi-level simulation approaches.

Domain-specific libraries

Dymola is based on Modelica which gives unique access to libraries developed by leading domain experts. All libraries are compatible with each other and include components for mechanical, electrical, control, thermal, pneumatic, hydraulic, power train, thermodynamics, vehicle dynamics, engine dynamics, air-conditioning, fuel cells, heat exchangers, etc.

Dymola Key Features

- Compliant with the Modelica® standard language, a powerful, object-oriented and formally defined modeling language.
- Comprehensive set of model libraries.
- Hardware-in-the-Loop (HIL) simulation in real-time on dSPACE, xPC and FMI platforms.
- Interface to FMI and Simulink®.
- Calibration and design optimization.

Customer testimonials

A number of Dymola applications have been described in journals and at conferences. A small

selection of papers recommended for further reading can be given here.

BMW has used Dymola for some 15 years, primarily for modeling conventional and hybrid drivelines [1]. Recently the Functional Mockup Interface has achieved great success as a simulation framework [2].

ZF Friedrichshafen uses Dymola and Modelica for modeling of transmissions, using the same models for MIL, SIL and HIL on a several different platforms [3].

SAAB Aeronautics uses Dymola for modeling of several vehicle systems, the most complex being the Environmental Control System and Fuel System of the Gripen multi-role combat aircraft [4] [5]. An interesting application is the use of CAD geometry data in system simulation [6].

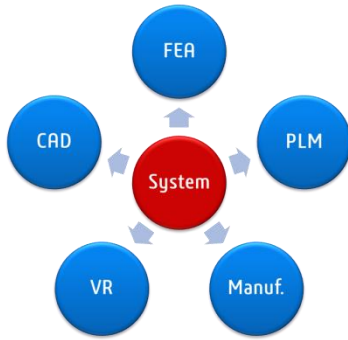
Ford Motor Company uses Modelica for several applications. The core effort has been focused on conventional and HEV drivelines [7] [8]. The challenges of making a full vehicle model run in real time are described in [9].

Toyota Motor Corporation has used Dymola for chassis, drivetrain (including the hybrid car Prius) and engine modeling [10]. More recently, a holistic vehicle model for small electric vehicles including mechanics, electrics, electronics, vehicle dynamics and control was made and utilized for the investigation of overall vehicle specifications and system structures [11].

Fluid and thermal simulation of complex systems is becoming more and more common in the automotive [12] and process industries [13].

Model Based System Engineering

A model based approach to product development is becoming more and more important, and Dymola plays an important role from conceptual design all the way to verification of the final product.

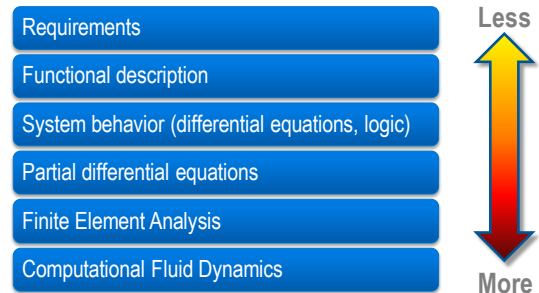


Every faith tends to place its deity at the center. The figure above shows how system simulation can interact with and provide services to many other domains.

- FEA: System-level simulation; boundary conditions.
- CAD: Dynamic simulation for mechanics and piping; import material properties.
- PLM: Requirements management for system models; collaboration tools. Simulation answering to quantitative requirements.
- Virtual Reality: Realistic behavior driving the virtual world.
- Manufacturing: Control and behavior for simulation.

Systems can be described at many levels of detail and complexity, starting with requirements specification going down to detailed mathematical models based on Finite Element Analysis (FEA) or Computational Fluid Dynamics

(CFD). Behavior modeling falls in-between, providing fast modeling and simulation at an intermediate level of detail.

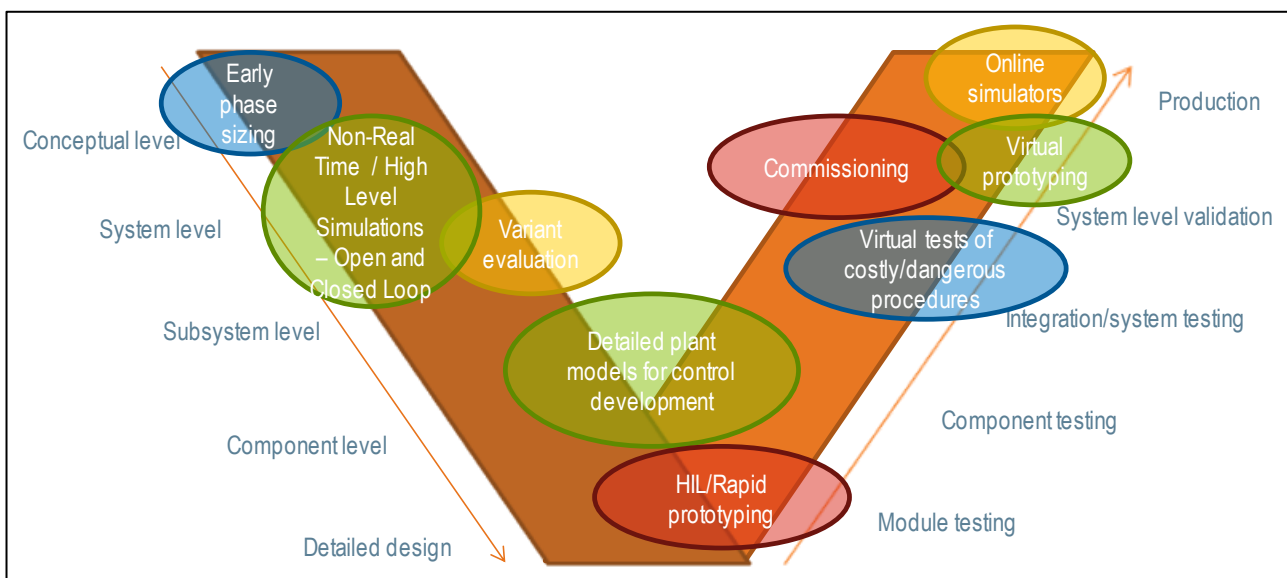


The development V

The traditional 'V' model can be used to visualize the various steps of the development process where system modeling and simulation can be deployed.

On the left hand, design, side of the V we descend from requirements capturing at the system and sub-system levels, then to architecture and sub-system design, and all the way down detailed component design.

On the right hand side, the models used on the design side are re-used during verification. Data obtained here is also used to calibrate the original models, leading to models that more accurately can predict the real-world behavior. Eventually, physical tests are used to verify that requirements have been satisfied.



The key benefits of a model based approach come from replacing physical prototypes with virtual prototypes, and the ability to evaluate design decisions earlier in the process. The overall goal is in fact to increase the number of prototypes, but virtually of course.

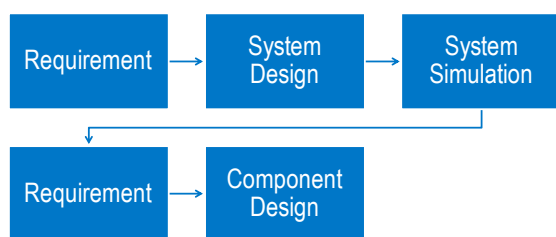
Early architecture assessment

Modelica supports template based modeling which makes it easy to experiment with alternative system architectures within a common test framework.

Very simple models can be used for early assessment, for example to estimate range and weight of Hybrid Electric Vehicle (HEV) designs in a wide range of scenarios. The evaluation of four different hybrid vehicles using a common model framework is described in [14].

Quantitative requirements

Certain system requirements are quantitative rather than qualitative. The most obvious example is fuel consumption for a given driving cycle, which will require a dynamic behavior model to estimate.

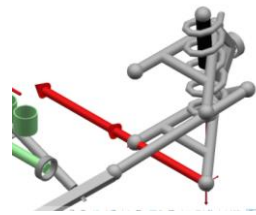


This is an example where an initial system requirement can be analyzed using simulation, leading to more detailed requirements for sub-system and component design.

Detailed subsystem design

More detailed behavior models are used to guide the design of sub-systems and individual components. An important aspect is that high-level models of the complete system can be augmented with detailed models of a smaller part.

Another possibility is to use system level simulation to calculate boundary conditions for Finite Element Analysis.



Relatively simple kinematic models of a wheel suspension can be used to calculate forces (red arrows) arising from a vehicle maneuver, which are then used for FEA of suspension parts.

Model verification

Moving up the right side of the development V, models used at the design stage are verified based on detailed simulation during component design, or by measurements on hardware.

This process is used to calibrate system level models so that they can more accurately predict system behavior. This knowledge is used if the design needs to be changed, or when system models are reused in a later project.

Hardware-in-the-loop (HIL)

The final testing of a system, before actual test driving of e.g. a car, is typically performed in a Hardware-in-the-Loop (HIL) test bench. Here one part of the system is implemented with the actual hardware, such as, an Electronic Control Unit (ECU). The rest of the system is implemented virtually, where a mathematical model emulates the rest of the car comprising of engine, transmission and vehicle.

HIL simulation imposes particular requirements, specifically that the simulation must run in real time to match the ECU. A typical cycle time for HIL applications is 1 millisecond. Dymola implements special optimization techniques to permit sophisticated drivetrain models to simulate reliably in a HIL environment.

Modelica

Standardization and vendor independence

Founded on the accumulated experience of several groups in industry and academia, Modelica is a modern and expressive language for describing dynamic systems. Particular attention has been placed on promoting true reuse of modeling knowledge regardless of application domain [15].

The Modelica language and its standard library (MSL) are managed by the Modelica Association through a number of design meetings. Modelica is vendor neutral and supported by multiple tools, being developed by both users and vendors.

Because Modelica is vendor and domain neutral, users are no longer locked into proprietary solutions. The open development process allows early feedback and yields well-defined semantics of the language.

Modelica Highlights

- Equation-based
- Acausal connections
- Declarative
- Object-oriented
- Template frameworks
- Non-proprietary

Equations and connections

Modelica is an equation-based modeling language. The dynamic behavior of systems is described by differential and algebraic equations. This allows tools to generate highly efficient simulation code, suitable for real-time Hardware-In-the-Loop applications.

At the lowest level of abstraction, relationships between variables are described by mathematical equations. At higher levels, components are composed graphically in Dymola thru drag-and-drop, and the externally visible variables (called

connectors in Modelica) are connected by drawing a line between two connectors.

A connection defines another set of equations, one for each variable in the connector. Across variables, such as voltage and force are set equal. Thru variables, such as current and torque, sum to zero.

Benefits of Equation-Based Modeling

Simple model definition

From textbooks we learn that an electrical resistor is defined by Ohm's law:

$$v = R \cdot i$$

We use this equation in the Modelica definition of a resistor too, here referring to the voltage drop between the positive and negative pins.

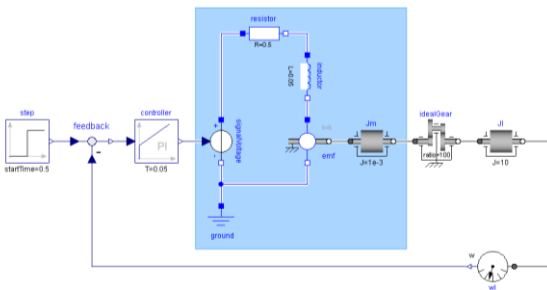
From this equation, a Modelica translator can either calculate the voltage drop (if the current is known), or by automatically re-writing the equation, calculate the current if the difference in voltage is known. Which is needed for each use of a resistor depends on the overall structure of the circuit. Modelica is said to be acausal, because the signal flow in and out of components need not be specified.

In tools which do not support equation-based modeling, the user must define two resistor models, one to calculate the voltage drop and one to calculate the current; for a component such as a planetary gearbox, many more combinations of inputs and outputs must be defined. Furthermore, the user must understand the detailed signal flow of the entire circuit in order to pick the right model for a component.

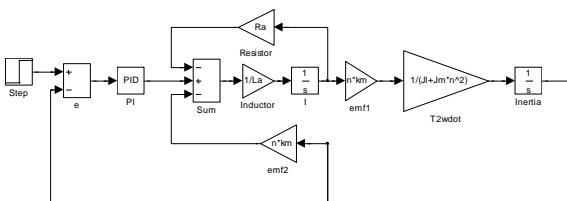
Dymola compared to block diagrams

The power of equation-based modeling compared to signal-based modeling can be shown with a simple example.

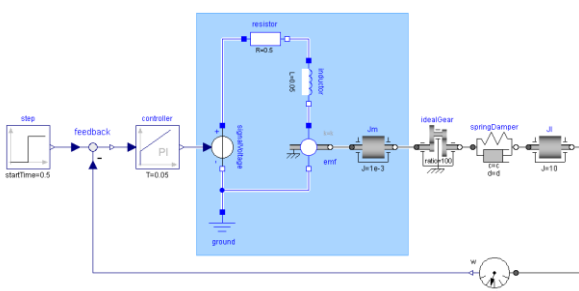
This Modelica model shows a simple servo, comprising a controller, an electrical motor (on blue background) and a load. The model has two inertias, one representing the rotor of the motor, and connected through an ideal gearbox, the load.



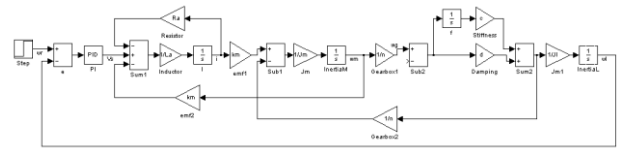
Such a model must of course be transformed into code suitable for simulation and execution. Dymola will do this automatically and efficiently, but using a traditional block-oriented tool the user must make this transformation by hand, which is both tedious and increases the risk of making mistakes.



Let us now study the vibrational effects of the shaft. We do this by inserting a spring-damper component in Dymola's diagram, which is a simple operation.



The corresponding operation in a block-oriented tool requires a major change of the model.

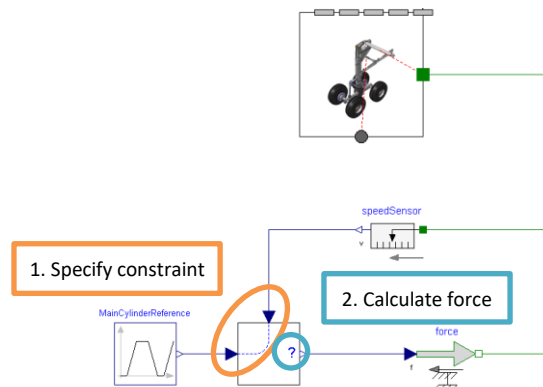


The key question is really if you want engineers to do this by hand, or if you want such work to be done automatically by a tool.

Inverse models

In an inverse model the simulation outputs are specified rather than the usual inputs, which are instead calculated by the model. The causality of the model is reversed, which requires an acausal model formulation.

A few examples of inverse models are a driver model that gives the perfect throttle setting to follow a given speed profile; calculation of actuator force needed for a mechanical assembly; and the design of model-based controllers.



[Model that calculates required actuator force to retract landing gear in specified time.]

In the example above we have designed a landing gear assembly. Previously we could specify a force that resulted in a motion of the mechanism. However, to dimension the system we want to prescribe a certain motion and calculate what forces are required to achieve this. The means we must invert the system.

With Dymola and an acausal Modelica model, we will instead specify the desired movement of the retraction connecting to the speed sensor. Dymola will then calculate the force needed to achieve this behavior. It should be noted that the landing gear model is unchanged.

Templates and Reuse

Interfaces and models

Modelica has been designed to provide very strong support for expressing common properties of models and interfaces that can be shared throughout an entire engineering domain. The Modelica Standard Library defines many such interfaces that are shared by commercial and third party libraries.

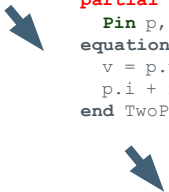
Inheritance

Inheritance is a composition mechanism inspired by object-oriented programming languages. It gives an efficient and safe construct to share common properties.

```
connector Pin
  Voltage v;
  flow Current i; // Sums to zero
end Pin;

partial model TwoPin
  Pin p, n; Voltage v;
  equation
    v = p.v - n.v;
    p.i + n.i = 0;
end TwoPin;

model Capacitor
  extends TwoPin;
  parameter Capacitance C;
  equation
    i = C*der(v);
end Capacitor;
```



[Reuse of models: Pin-TwoPin-Capacitor.]

In this example, we start by defining an electrical pin with two variables: voltage and current. This definition of an electrical pin can be reused by any electrical component, ensuring connection compatibility.

The pin is then used to define a TwoPin, the common basis for all electrical components with two pins. Furthermore, we can define the basic

property that current flowing into one pin will flow out of the other pin, as well as a variable for the voltage drop.

Based on this TwoPin, the definition of electrical components such as a capacitor can be reduced to the characteristic equation, reusing all interface specifications.

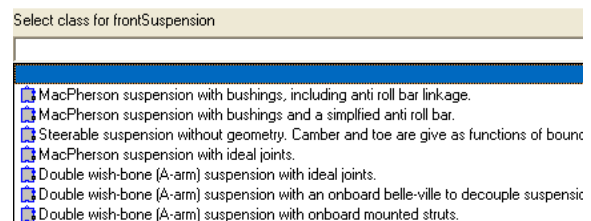
Model templates

Modelica supports the development of model templates, where certain parts of a model can be replaced by compatible components.

The author of the template must declare what parts can be changed. Typically the author also specifies a model which defines the interfacing requirements of any substituted component. The user of a template can then redeclare components in the template, thereby providing the behavior model for a specific part.

Model templates provide a clean interface between template author and template user. Compared to a copy-and-paste approach, Modelica templates ensure compatibility and make explicit which components are modified.

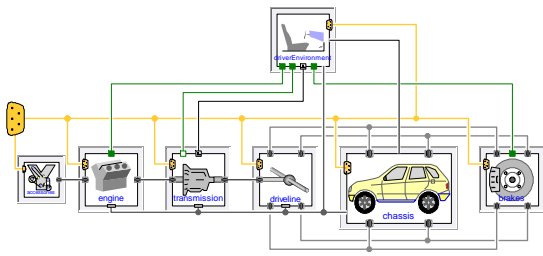
The graphical user interface in Dymola facilitates the use of templates. Dymola can list exactly those models which are interface compatible, making selection easy.



[Changing the front suspension implementation from a list of matching models.]

Reusable frameworks

The vehicle architecture is a powerful example of a Modelica template, which gives a framework for many different cars.



[Standard vehicle architecture from the VehicleInterfaces library.]

The template defines an architecture comprised of engine with front-side accessories, transmission, driveline, chassis and breaks, as well as a driver model. Each sub-system can be replaced by more detailed components that model specific technical solutions, down to models parameterized to represent manufactured parts from suppliers.

The framework ensures component compatibility, convenient default behavior, and a common signal naming convention that facilitates experimentation and reuse across department boundaries.

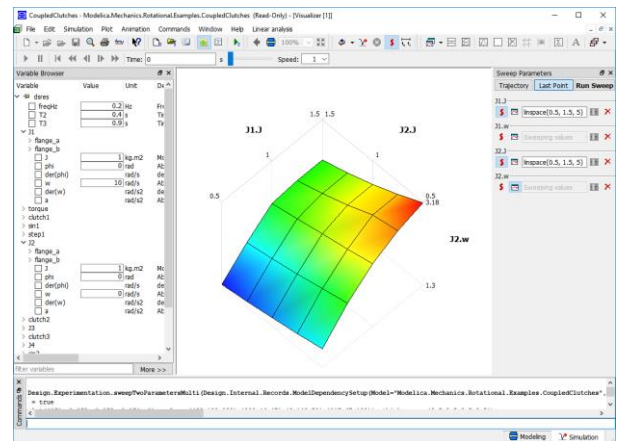
Dymola Model Reuse
Primitive component models are reused to 75-90 %. Reuse at the system model level (250-300 state variables) is approximately 40-60 %. Reusing model interfaces with 300-400 in/out signals saves a lot of work.
Ingela Lind, PhD Technical Fellow, SAAB Aeronautics

Model Design Tools

Sweeping parameters

Model experimentation involves running simulations for various combinations of parameters to determine the modeled system's properties.

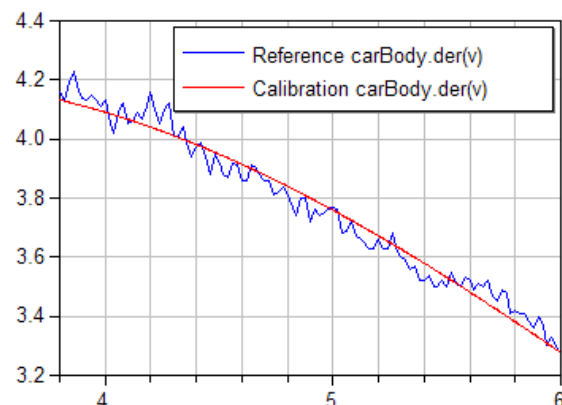
A convenient user interface facilitates such experimentation. Sweeps over one or two parameters are visualized by plotting variables used as key performance indicators, or the surface formed by varying two parameters.



[Sweeping of two parameters visualized by a surface plot. Experiment setup on the right side.]

Model calibration

A Modelica model describing a physical system typically includes many parameters which have to be set. Some parameter values are difficult to determine from the design specification or hard to measure, for example the inertia of a part, friction and loss parameters.



[Comparing measured and simulated acceleration after tuning model parameters.]

Model calibration (parameter estimation) is the process where measured data from a real device is used to tune parameters such that the simulation results are in good agreement with the measured data. Dymola varies the tuning parameters and simulates to search for satisfactory solutions which minimize the difference between the simulation results and the measurements.

Design optimization

The Design Optimization option is used to tune parameters of a device or its controller to improve system dynamics for multiple criteria and multiple cases.

A Modelica model contains many parameters that can be tuned for better performance, for example, the spring constants of a car, the gear ratio of a gearbox, or parameters of a controller.

Design optimization is an approach to tune parameters such that the system behavior is improved. The tuning parameters are calculated to minimize mathematical criteria which express improvement. Criteria values are usually derived from simulation results, e.g., the overshoot or rise time of a response, but they can also be derived by frequency responses or eigenvalue analysis.

Model management

Model Management includes support for encryption of models, version control from Dymola (CVS and Subversion) and utilities for checking, testing and comparing models.

- Regression testing (checking simulation results against known good results).
- Class and condition coverage.
- Variable unit and style checking.

Code and Model Export

Dymola has support for exporting models and model source code. Three export alternatives with different functionality are provided.

Real-time simulation

The real-time simulation option enables the model to be used in environments not supporting the Microsoft C compilers. The option is specifically designed for real-time platforms, such as the dSPACE, xPC and other platforms that are supported by Dymola for Hardware-In-the-Loop (HIL) simulation.

Real-time simulation allows export of models that use inline integration, a methodology that embeds fixed-step numeric integration in the model code to improve simulation performance.

Binary model export

The binary model export option allows the model to be exported to other computers without requiring a Dymola license at the target system. The simulation functionality of the exported model is the same as on a computer having a Dymola license.

Source code generation

Source code generation exports code that can be used on any platform without the need of a Dymola license at the target system. It allows export of readable and well-documented code facilitating inspection, debugging, profiling, etc. This makes this export option suitable for advanced model-based applications, such as rapid prototyping.

The binary model export and source code generation options both allow export of symbol table information, e.g., model structure, variable names, types, and units as an XML file.

Model encryption

Models may contain sensitive information, either design parameters or in the model formulation. Hence there may be a need for the model author to protect the contents from copying or viewing.

Dymola supports encryption of packages of models, and has settings to hide or show certain aspects of the model. For example, the documentation of a model can be visible, but its implementation hidden. However, the internal structure of the model can be used by the tool, but the generated simulation code will contain obfuscated variable names to reduce the risk of reverse engineering from the C code.

For complete sub-systems, the executable model can be exported as binary code according to the FMI standard.

Interfacing Other Software

Dymola offers excellent capabilities to interface to other software tools and simulation environments.

Functional Mockup Interface

FMI allows any modeling tool to generate C code or binaries representing a dynamic system model which may then be seamlessly integrated in another modeling and simulation environment. Dymola supports import and export of functional mockup units in all formats and full compliance with the FMI specification.

Dassault Systèmes provides tools with full support for FMU export and import with Simulink.

Native Simulink support

In addition to FMI, Dymola also supports export of S-function blocks for direct integration into the Simulink environment. The tool chain is fully compatible with HILS platforms such as dSPACE.

Python, Java and JavaScript

Dymola can easily interface to common scripting environments such as Python and Java, permitting flexible scripting of common tasks. Parameters can be set and simulation results read with provided utility libraries. Data can be exported in various formats, including CSV for Excel and HDF5.

Academic Packages

Modelica-based Dymola offers a common platform for teaching across many scientific domains. Focus on equations from physics, chemistry or mathematics rather than implementation of algorithms. Apply industrial strength model libraries to teach applications that are more interesting. There are a number of

favorably priced packages for academic use, with either basic functionality or including all options.

Learn and innovate

The main academic packages are called Learn and Innovate. The Academic Learn package (DYL-EDU) contains the standard Dymola configuration, a few libraries, FMI export capability plus the options to use models in Simulink and on real-time platforms.

The Academic Innovate package (DYI-EDU) includes all commercial Modelica libraries in addition to the features of Academic Learn.

Teaching offer

There is a special offer for teaching, which comprises 25 shareable licenses of either Learn or Innovate at a special rate. The motivation being that a typical classroom has up to 25 students.

Student licenses

The student licenses (available as Learn and Innovate) are intended for use on the student's own computer for home usage, whereas the usual academic licenses are installed on the university's computers. The student licenses are have reduced capacity (only smaller models can be translated) and are valid for one year.

User Roles

Development engineer

The development engineer works on new designs as well as improving existing ones. He wants to focus on modeling the system behavior, both of the physical sub-system that will be manufactured and of the associated control system. To do that he wants an environment where he can focus on the system design and the model behavior, and care less about programming.

Dymola provides an effective, Modelica based, environment for model development, simulation

and experimentation. Most component models are readily available in libraries, composed with a simple drag-and-drop operation and then parameterized. Specialized models can be developed from scratch using the open Modelica language.

Using an equation-based Modelica solution means that the development engineer does not have to manually transform his models to simulation code, this is handled automatically by Dymola. This saves time, because the translation is fast and automatic, and it increases reliability because human error is eliminated in the process. Development becomes faster and more robust, giving him time to explore more alternatives and changing his design without having to create simulations from the beginning again. When hardware testing begins, models can be run on many common Hardware-in-the-Loop platforms.

He will have more fun and be more productive. Because he can capture more complicated behavior early in the modeling work, the designed product will be better suited for the user's needs.

Project manager

The project manager has a group of development engineers designing a range of advanced products. One of the key challenges is to make developments by one engineer available to the entire group, and eventually to other development groups responsible for system integration.

By developing libraries of component models in the team, commonality of design and alternative solutions can be expressed. This leads to higher productivity due to reuse of models from earlier development work. Developing common test models gives a framework that makes exploration of alternative designs easy.

Models are used in the entire development cycle, from exploring system architectures, through detailed design, and finally to verification and HIL testing.

The project team must also support other departments developing connected sub-systems and a group responsible for system integration. They do this by exporting pre-packaged simulation code according to Functional Mockup Interface (FMI) specification, which protects the model integrity but makes integration with other simulation tools easy.

HIL test engineer

The manager of a Hardware-in-the-Loop (HIL) laboratory is responsible for testing real hardware components in a virtual test environment. The goal is "test more prototypes, not fewer" but obviously not meaning to build more full-scale hardware prototypes.

The HIL lab is designed to test certain hardware components, typically controller ECUs, together with the rest of the car emulated by software. This approach is less expensive than building physical prototypes, shortens the time to complete a test, and gives reproducible test conditions. For a powertrain ECU, the transmission, driveline, parts of the car, and the drive cycle are implemented in software. To support it a dedicated real-time computer system with data acquisition is needed. The tool used for modeling the transmission must also be able to generate code that runs in real time.

Dymola is the Modelica tool most commonly used for HIL applications. While maintaining the high-level model description inherent by Modelica, Dymola uses specialized techniques in order to generate C code that is well suited to running in real time on HIL platforms. The same models can be used for both HIL and non-real time models. This means that models developed during the design phase (the left hand side of the development V) are used also for verification (the right hand side of the V). Also measurement data from the HIL environment can be used to further improve the original models, using a process called model calibration.

IT manager

The IT manager working in a large corporation has to support multiple user groups with different needs. There are researchers, system developers, and large sets of simulation data with associated models. Many different tools are used.

Typically, models are developed within a department, and then the simulation code needs to be shared by several users in other departments. Model integrity, and sometimes hiding of sensitive information, is essential. This creates the need for a common simulation and execution environment, where simulation modules can be used in a cross-department framework of computers.

For these reasons, many companies have decided to deploy the Functional Mockup Interface (FMI) as standard for the simulation infrastructure. FMI is well specified and vendor-neutral, allowing the use of multiple tools for developing simulation models and also several tools to execute the models on various platforms.

VP R&D

The head of R&D has an overall strategic responsibility for all development work. To ensure a long-term stable environment, the goal is to focus on open, standardized, vendor-neutral formats supported by multiple tool vendors: the Modelica language for cross-domain model development, and the Functional Mockup Interface (FMI) for simulation execution and exchange.

To sustain a steady flow of innovation, the collected knowhow of current and previous systems encoded in Modelica models, which allows global reuse of engineering knowledge. Reusable models are used from early architecture studies, through detailed design of a smaller number of proposed solutions, all the way to system verification and validation.

Using common modeling and simulation platforms supports cross-department collaboration, exchange of human resources and transfer of knowhow from one generation of engineers to the next.

Professor

As professor at an engineering school, the difficulty is to combine the teaching of principles with relevant examples. Without the proper tools, examples worked out by hand tend to be simplistic and without industrial relevance. With the wrong tools, too much time is wasted on transforming the problem to code rather than engineering.

With the unified modeling approach given by Modelica, modeling based on first principles across multiple domains can be expressed directly with equations, and the powerful Dymola tool takes care of generating simulation code. Because Modelica can be used in many domains, the cost of teaching the tool is amortized over the several courses.

With special academic licensing, including all commercial model libraries, Dymola is a very affordable modeling and simulation platform. To facilitate teaching, license bundles for classroom or individual student use are available.

Industry Solutions: Model Libraries

Modelica tools are domain-agnostic, meaning that they process equations in order to generate efficient simulation code. Domain knowledge is packaged in Modelica model libraries, designed to handle a variety of applications from mechanical, electrical, and thermo-fluid domains.

The libraries can be used with both Dymola and **3DEXPERIENCE** Dymola Behavior Modeling, either on their own or combined with other Modelica

Libraries, to model and simulate complex systems that can span multiple engineering disciplines.

Automotive

The automotive applications fall into three main categories. The engine and drive train are modeled using the Vehicle System Modeling and Analysis (VeSyMA), VeSyMA Engines and VeSyMA Powertrain libraries. The flexibility of the open Modelica language is particularly suitable for modeling hybrid or alternative drive trains using the Battery, Brushless DC Drives and Electrified Powertrains libraries. Modal bodies or flexible shafts are available through the Flexible Bodies library. Engine and battery cooling is supported by the Cooling library, which can be combined with the Thermal System Simulation and HVAC libraries. The Human Comfort and Fluid Dynamics libraries adds models of occupant comfort for complete vehicle thermal modeling.



Subsystem and complete vehicle models for handling and control experiments are provided by the VeSyMA Suspensions library, which also allows real-time simulation for driver-in-the-loop applications. The hierarchically structured, open-source, Modelica models offer unprecedented flexibility for multiple vehicle configurations while reusing common components.

Actuator and controller components are available in the Fluid Power and Pneumatic Systems libraries and the Modelica Standard Library.



Aerospace and defense

A multitude of libraries offers the capacity to model the complex thermo-fluid systems of aircraft, ranging from fuel systems to environmental control. A wide range of thermo-fluid systems can be modeled with the Thermal System Simulation library. The Human Comfort library provides additional models of occupant comfort for cabin thermal modeling.

The Flight Dynamics Library enables rapid analysis of flight characteristics of a wide range of flight vehicles. The library is ideally suited for the multi-disciplinary development of accurate flight control laws as well as for use in real-time flight simulators.

Actuators for flight control and other subsystems use the Fluid Power, Pneumatic Systems, Brushless DC Drives and Electrified Powertrains libraries. Flexible beams and modal bodies from Finite Element models are managed by the Flexible Bodies library.

The Electric Power Systems library supports modeling of electrical aircraft systems, including generation, conversion and control of high frequency A/C systems.



Systems, Battery, Brushless DC Drives and Electrified Powertrains libraries. The thermal properties of industrial machinery are easily modeled with the Thermal System Simulation and Cooling libraries.

Energy, process and utilities

Ever more stringent requirements on environmental impact drive the trend towards more detailed modeling of physics and control systems. The ClaRa Plus library facilitates simulation of e.g. advanced combined cycle power plants.

The Heat, Ventilation and Air Conditioning (HVAC) library allows you to minimize building HVAC operating costs by selecting the correct system control strategy and avoid costly HVAC system design errors early in the building design process. The Human Comfort Library provides an integrated approach to simulate the thermal comfort within an occupied building or vehicle.



Industrial equipment

All kinds of industrial equipment can be modeled using the mechanical libraries of the Modelica Standard Library, including 3D multi-body systems. Other options are flexible beams and modal bodies originating from a Finite Element model. Actuators and control systems are modeled with the Fluid Power, Pneumatic

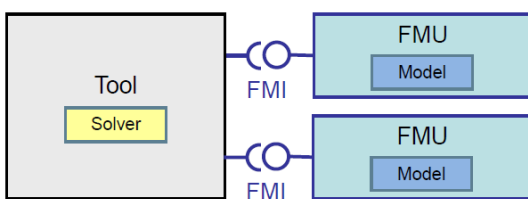
Functional Mockup Interface

The Functional Mockup Interface (FMI) is an industry standard for combining simulation code modules (FMUs) from multiple tools and vendors. Developed under the auspices of the Modelica Association, the specification provides a well-defined and vendor-independent exchange format for code (binary or source) as well as associated data and documentation.

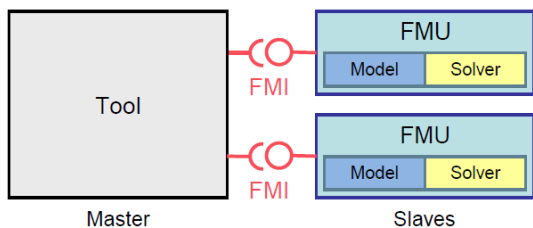
FMI is supported by a large number of authoring tools, including tools which are not Modelica based, making it the ideal foundation for a vendor independent simulation infrastructure.

The FMI specification defines two exchange formats. FMI for model exchange defines the interface for simulation code modules that must be combined with a common, central, solver. This ensures a uniform numeric solution and centralized strict simulation error control.

FMI for co-simulation defines the interface for code modules with embedded numeric solvers, as used by the generating tool. This approach gives the opportunity to embed dedicated solvers for the modeled application, and facilitates compatibility with simulation in the authoring tool.



[FMI for Model Exchange.]



[FMI for Co-Simulation.]



The recent FMI 2.0 specification standardizes important extensions that improve simulation speed and accuracy.

FMI 2.0 Highlights
<ul style="list-style-type: none"> • Classification of interface variables • Save and restore FMI module state • Variable dependency information • Partial derivatives • Precise time event handling • Improved unit definitions

FMI tools for Simulink – FMI Kit

Dassault Systèmes provides tools with full support for FMU export and import with Simulink. The toolkit can be used for free without any license key. Support and maintenance is offered to Dymola customers through the regular support channel.

General features of the FMI Kit are:

- Full FMI support for both export and import; FMI versions 1.0 and 2.0 - Model Exchange and Co-Simulation.
- Simulink Coder Target is used for export of FMUs from Simulink.
- Support for global tunable parameters and parameter references to workspace or mask variables.
- Variable communication step size in Co-Simulation export.
- Simulink FMU block for importing and embedding FMUs into Simulink models.
- Import of Dymola source code FMUs and support for several simulation targets: Rapid Accelerator, RSIM, GRT, and dSPACE DS1005 and DS1006.

References

Public sources of additional information.

- [1] C. Schlegel, M. Bross and P. Beater, "HiL-Simulation of the Hydraulics and Mechanics of an Automatic Gearbox," in *Proc. 2nd Modelica Conference*, Oberpfaffenhofen, Germany, 2002.
- [2] S.-A. Schneider, J. Frimberger and M. Folie, "Significant Reduction of Validation Efforts for Dynamic Light Functions with FMI for Multi-Domain Integration and Test Platforms," in *Proc. 10th Modelica Conference*, Lund, Sweden, 2014.
- [3] J. Köhler, M. Kuebler and J. King, "Transmission Modeling in Modelica: A consistent approach for several software development platforms," in *Proc. 10th Modelica Conference*, Lund, Sweden, 2014.
- [4] I. Lind and H. Andersson, "Model Based Systems Engineering for Aircraft Systems – How does Modelica Based Tools Fit?," in *Proc. 8th Modelica Conference*, Dresden, Germany, 2011.
- [5] S. Steinkellner, H. Andersson, H. Gavel and P. Krus, "Modeling and simulation of Saab Gripen's vehicle systems," in *Proc. AIAA Modeling and Simulation Technologies Conference, AIAA 2009-6134*, Chicago, IL, USA, 2009.
- [6] I. Lind and A. Oprea, "Detailed geometrical information of aircraft fuel tanks incorporated into fuel system simulation models," in *Proc. 9th Modelica Conference*, Munich, Germany, 2012.
- [7] M. Tiller, W. E. Tobler and M. Kuang, "Evaluating Engine Contributions to HEV Driveline Vibrations," in *Proc. 2nd Modelica Conference*, Oberpfaffenhofen, Germany, 2002.
- [8] J. Batteh, "Engine Modeling with Modelica," in *Proc. Modelica Automotive Workshop*, Dearborn, MI, USA, 2002.
- [9] R. Gillot, A. Picarelli, M. Dempsey and S. Gallagher, "Model Reduction Techniques Applied to a Physical Vehicle Model for HiL Testing," in *Proc. 12th Modelica Conference*, Prague, Czech Republic, 2017.
- [10] S. Soejima, "Examples of usage and the spread of Dymola within Toyota," in *Proc. Modelica Workshop*, Lund, Sweden, 2000.
- [11] Y. Hirano, S. Inoue and J. Ota, "Model-based Development of Future Small EVs using Modelica," in *Proc. 10th Modelica Conference*, Lund, Sweden, 2014.
- [12] E. Galindo, R. Soler, A. Picarelli and V. Avila, "Engine thermal shock testing prediction through coolant and lubricant cycling in Dymola," in *Proc. 12th Modelica Conference*, Prague, Czech Republic, 2017.
- [13] T. Skoglund, "Simulation of Liquid Food Processes in Modelica," in *Proc. 3rd Modelica Conference*, Linköping, Sweden, 2003.
- [14] J. Batteh and M. Tiller, "Implementation of an Extended Vehicle Model Architecture in Modelica for Hybrid Vehicle Modeling: Development and Applications," in *Proc. 7th Modelica Conference*, Como, Italy, 2009.
- [15] H. Elmqvist, "Modelica Evolution - From My Perspective," in *Proc. 10th Modelica Conference*, Lund, Sweden, 2014.