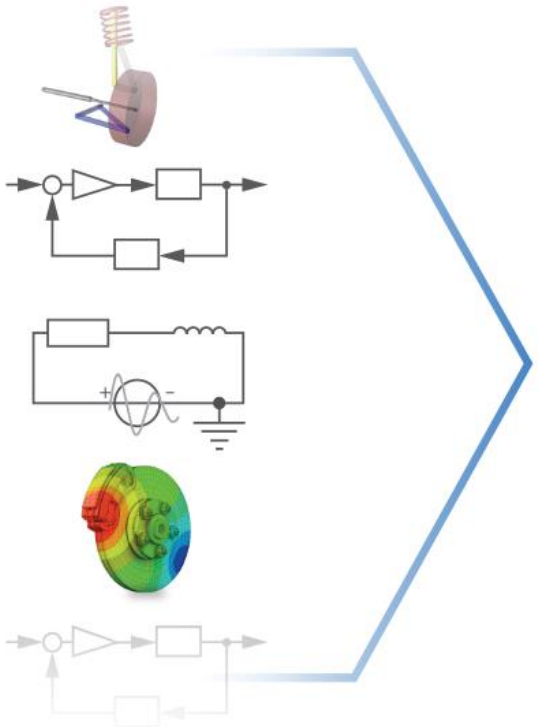
Status & Updates January 2021

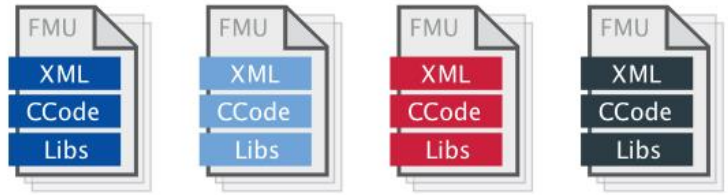
Contents

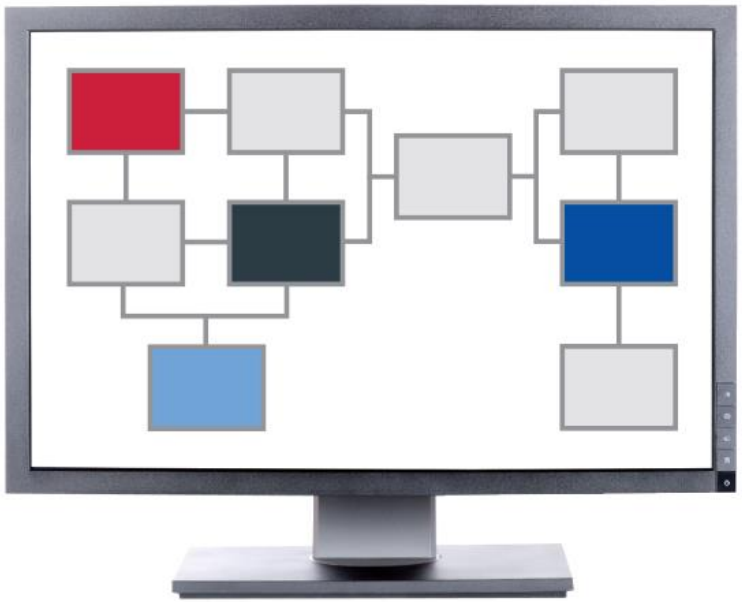
- Overview
- Quick Overview of Modelica Association Standards
- New in FMI 3.0
  - FMI Interface Types (Model Exchange, Co-Simulation, Scheduled Execution)
  - FMI for Co-Simulation
    - Event Handling
    - Intermediate Update
  - Support of Layered Standards
  - Miscellaneous
- eFMI: FMI for embedded systems, current status

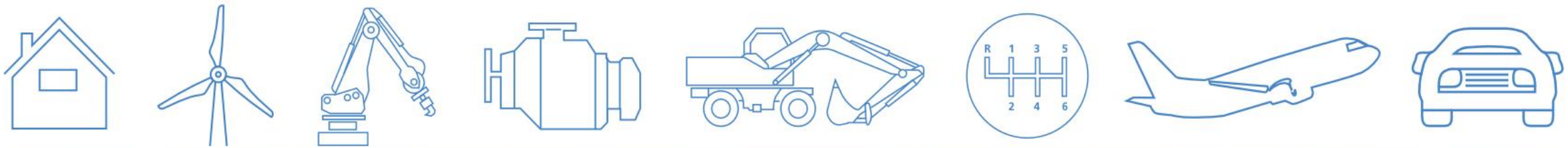
Overview



fmi Functional Mock-Up Interface





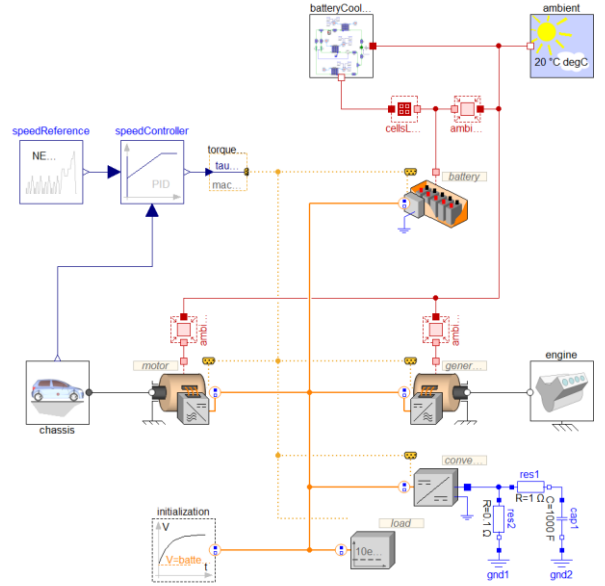


Modelica Standards for Engineering Simulation

Non-profit Modelica Association:

- Provides a family of coordinated standards for the modeling and simulation of cyber-physical systems.
- All standards definitions are available free of charge to anybody
- Development done in open groups of interested parties

- A-causal, object-oriented language & free libraries for physical systems modeling and simulation.
- Transient & steady-state simulation and optimization



Modelica Standards for Engineering Simulation

Modelica Language

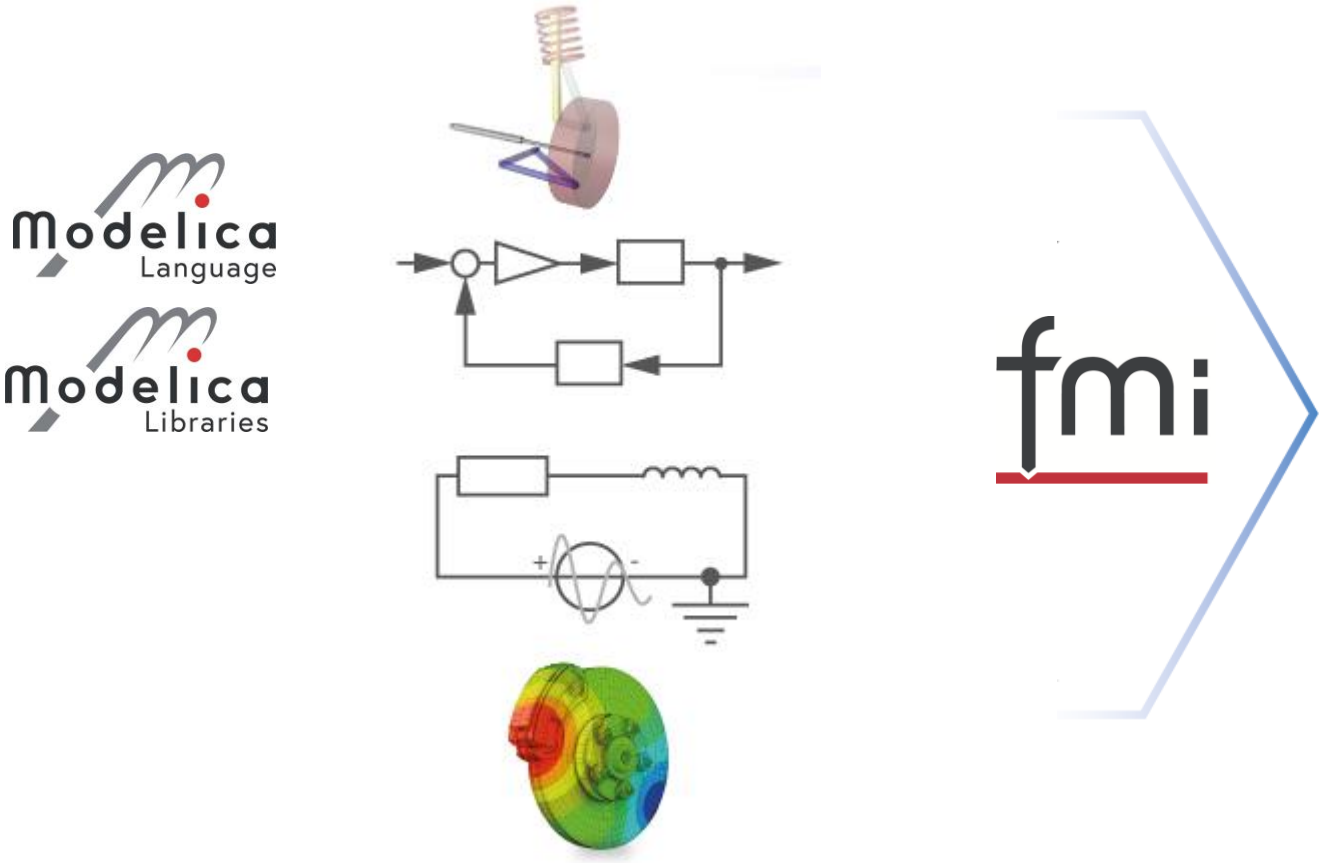
- First Release, Modelica 1.0: 1999
- Current Release, Modelica 3.4: 2017
- Modelica 3.5 to be released in February 2021

Modelica Standard Library

- Current release: Modelica Standard Library 4.0, released June 2020

FMI: Functional Mock-Up Interface

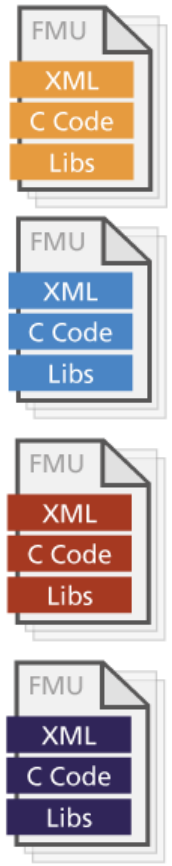
- Semantic, API and XML format for exchange of simulation models and co-simulation

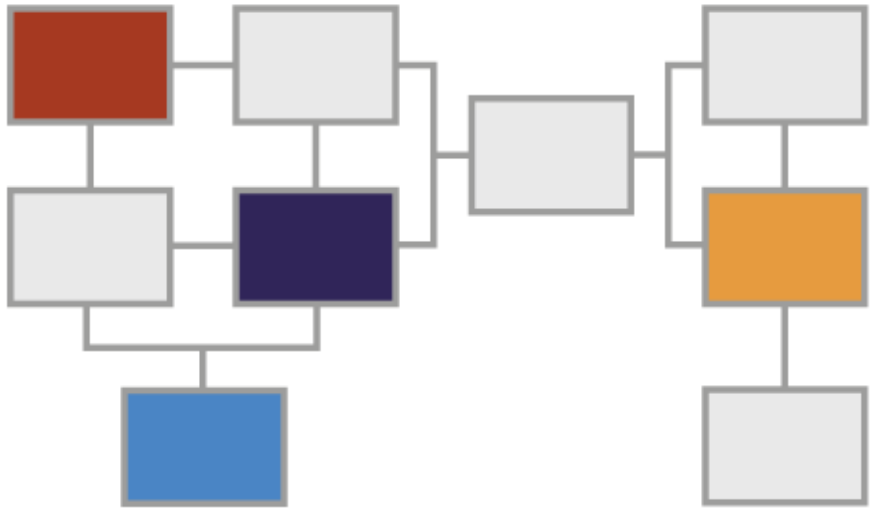


https://fmi-standard.org/

SSP: System Structure and Parametrization

- File format and XML schemas for description of architectures and sets of parameters





ssp System Structure & Parameterization

https://ssp-standard.org/

Modelica Standards for Engineering Simulation

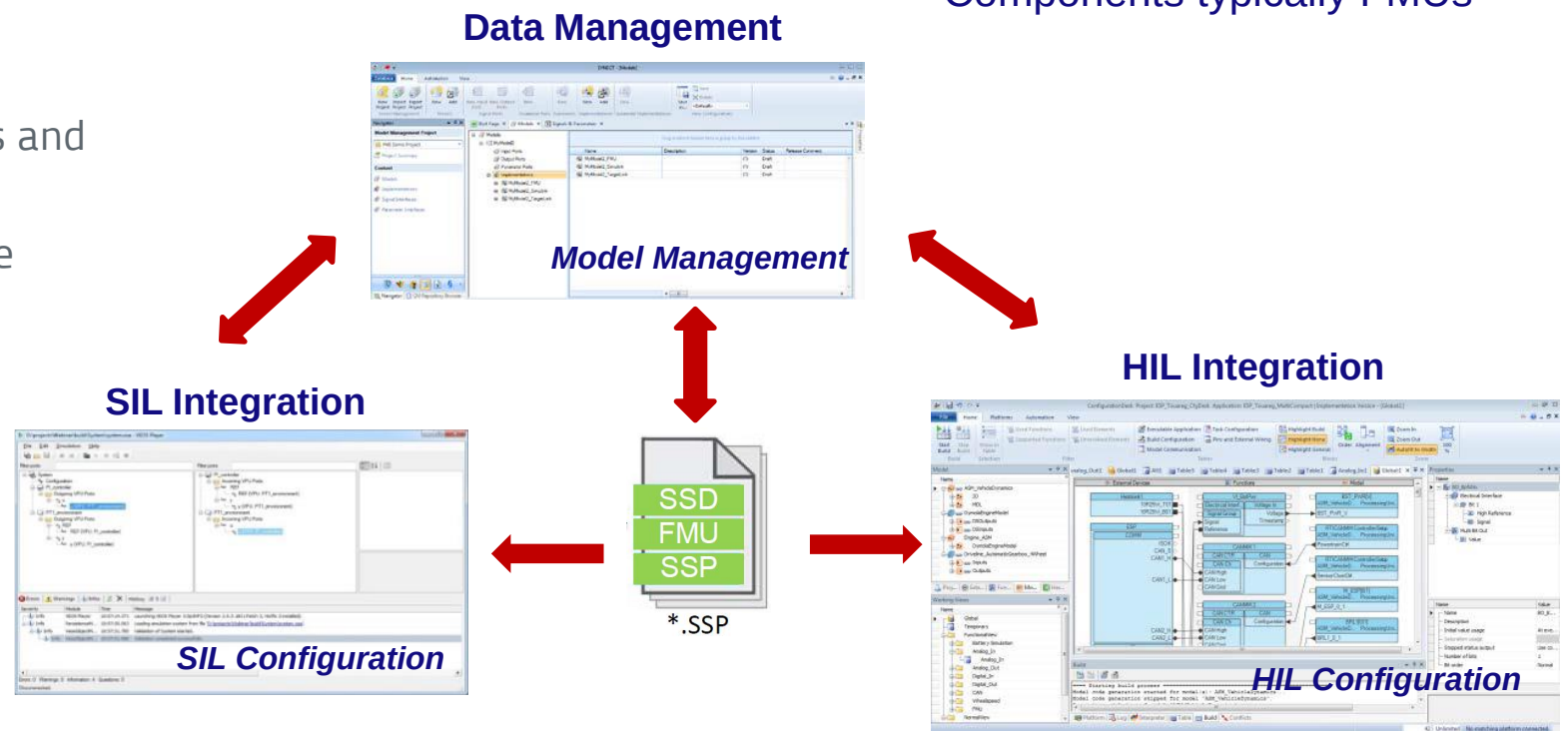
Integration of FMUs for SIL & HIL

- The System Structure defined for SIL can be reused for HIL testing
- It becomes possible to reuse more models, configurations, tests, layouts and parameters
- A Data Management tool controls the lifecycle of the SSP

Release 1.0: March 2019
Status: update for best compatibility with FMI 3.0

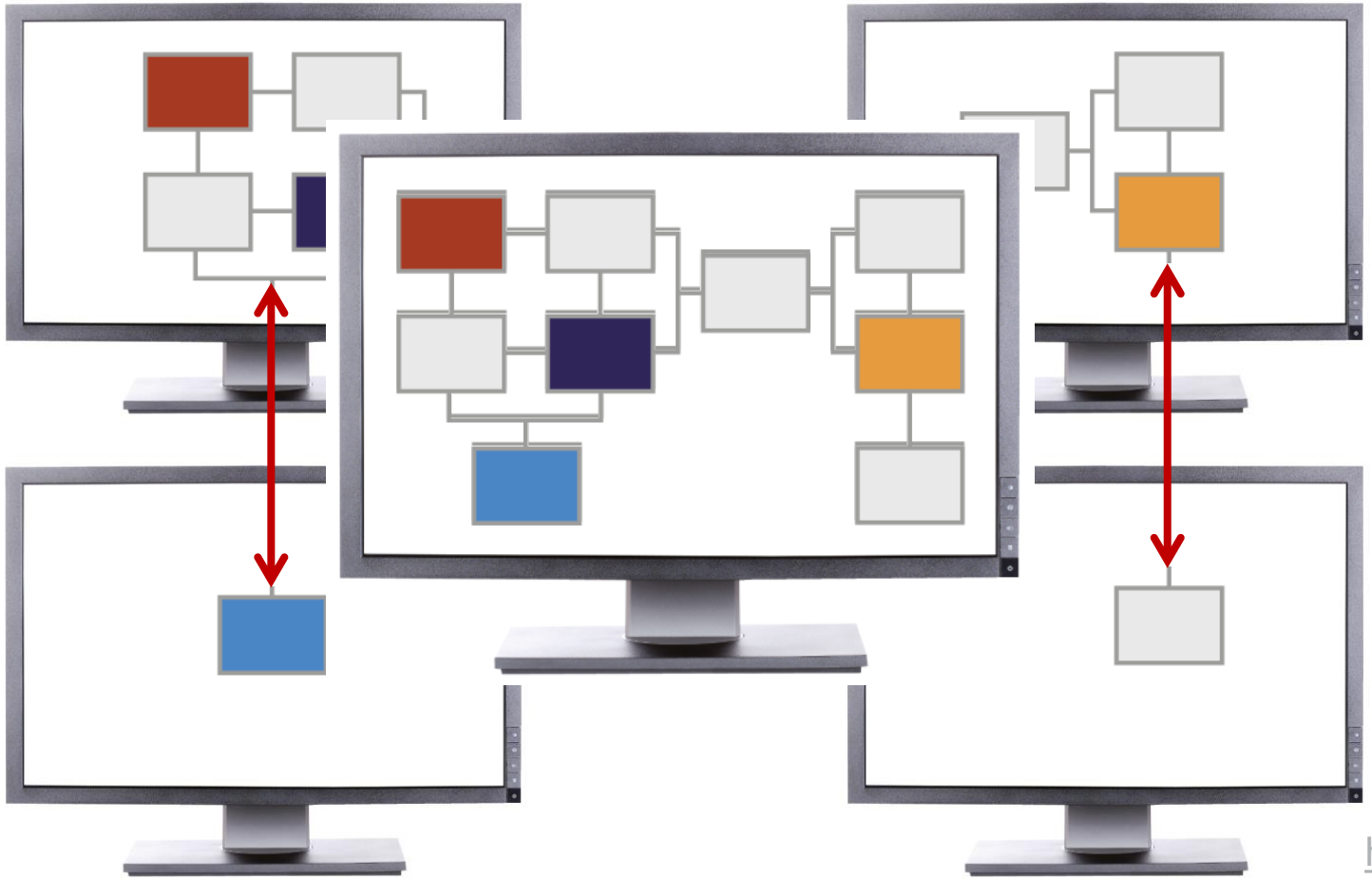
SSP System Structure & Parameterization

- Describe system configuration, structure, parameters for composite cyber-physical systems
- Components typically FMUs



DCP: Distributed Co-Simulation Protocol

- Semantic, protocol and XML schema for network-based co-simulation



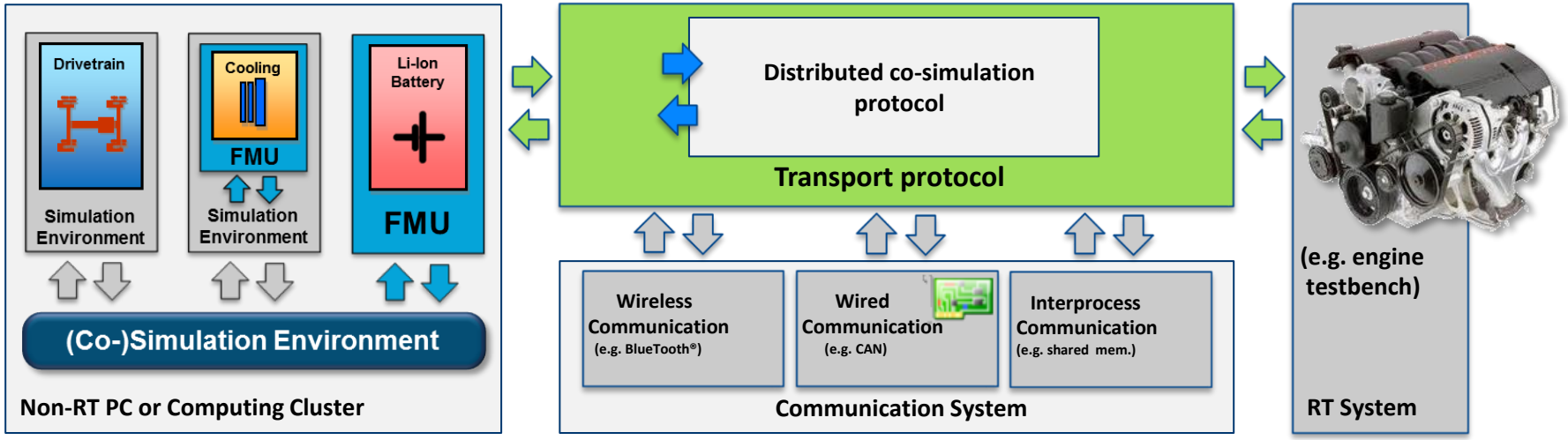
dcp Distributed Co-Simulation Protocol

https://dcp-standard.org/

Modelica Standards for Engineering Simulation

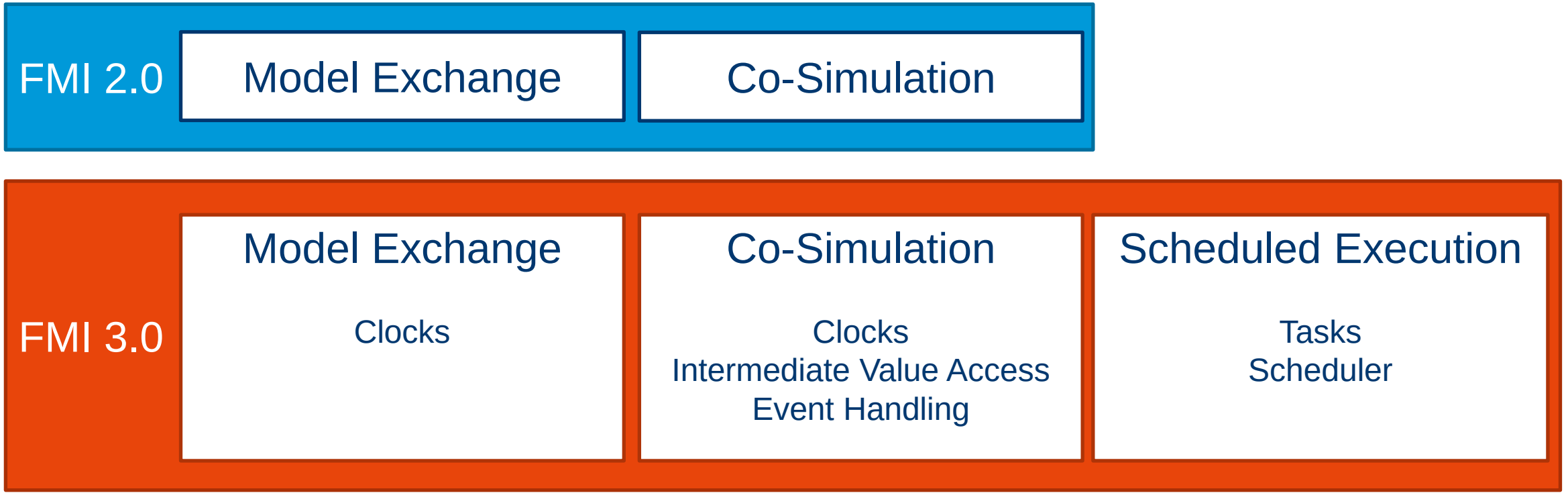
dcp Distributed Co-Simulation Protocol

- Real-time communication protocol between distributed models
- For HiL simulations and communication between models and real systems



Release 1.0: March 2019

FMI 3.0: Interface Types



NEW FEATURES FOR CO-SIMULATION

New FMI 3.0 Feature	Co-Simulation Use Case
Events (raising and handling)	Interrupt doStep() to handle events
Intermediate Update of Input Variables	Better interpolation of inputs to improve stability and efficiency
Clocked systems	Synchronize variable changes across FMUs

NEW FEATURES TO SIMPLIFY HANDLING

New FMI 3.0 Feature	Use Case
Array Support	Simplify usage of multi-dimensional variables
Terminals and Icons	Simplify connection of many signals between FMUs in importers

NEW FEATURES FOR CONTROLLER PACKAGING (VECU)

New FMI 3.0 Feature	vECU Use Case
integer types and Float32	vECU internals and ***bus*** signals (8, 6, 32 and 64 bit, signed and unsigned integer, 32 and 64 bit float)
binary type	Complex sensor data and ***bus*** signals
clocked variables	***Bus*** signals
array support	Internal maps
structural parameters	Calibrate array sizes at runtime
terminals and icons	Group ***bus*** signals
clocked partitions	External scheduling of tasks
new interface type: Scheduled Execution	

NEW FEATURES FOR LAYERED STANDARDS

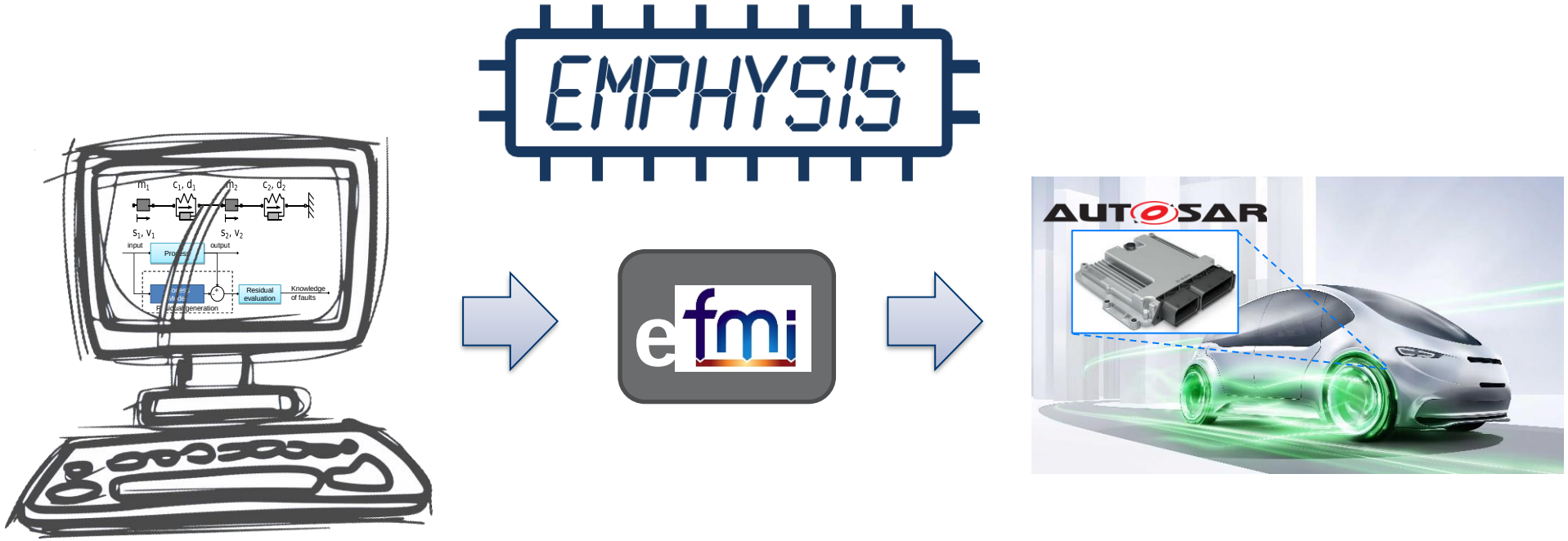
New FMI 3.0 Feature	Layered Standard Use Case
XML element annotations	Semantic description of variables and terminals
/extra folder in zip	Ship additional files required

Layered Standards (in preparation)	Use Case
XCP	When packaging vECUs (controllers), XCP allows standardized access (see ASAM) to ECU internal variables
CAN2Signals	Allows grouping and description of FMU IO as CAN bus signals (and possibly binary frames)

Standardizing eFMI for Embedded Systems with Physics-based Models in the Production Code Software

eFMI Short Summary

Jan. 29, 2021, Germany



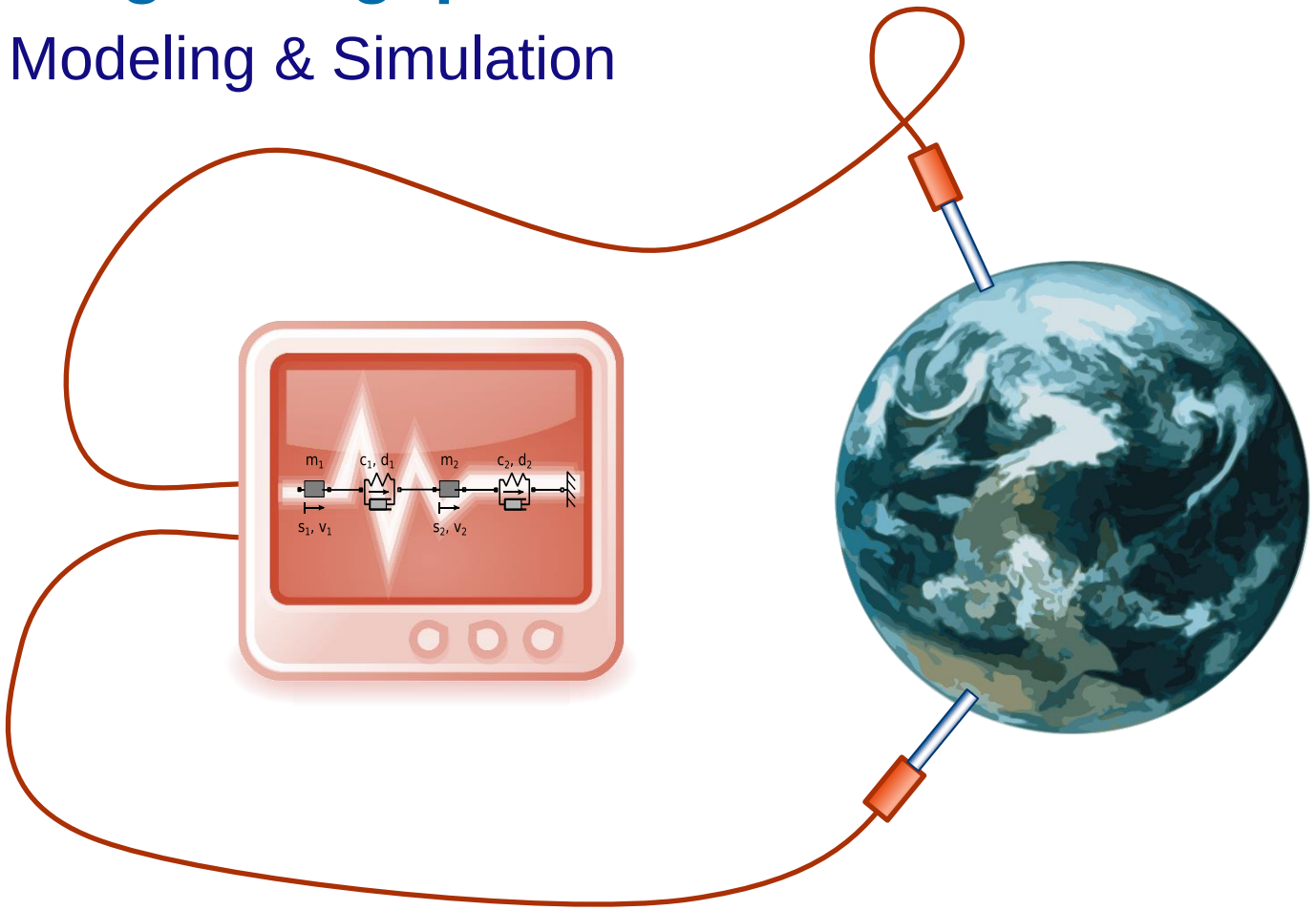
Oliver Lenord, Robert Bosch GmbH – Corporate Research
Martin Otter, DLR
with contributions from all partners of the EMPHYSIS consortium

SPONSORED BY THE
Federal Ministry of Education and Research
VINNOVA
AGENTSCHAP INNOVEREN & ONDERNEMEN

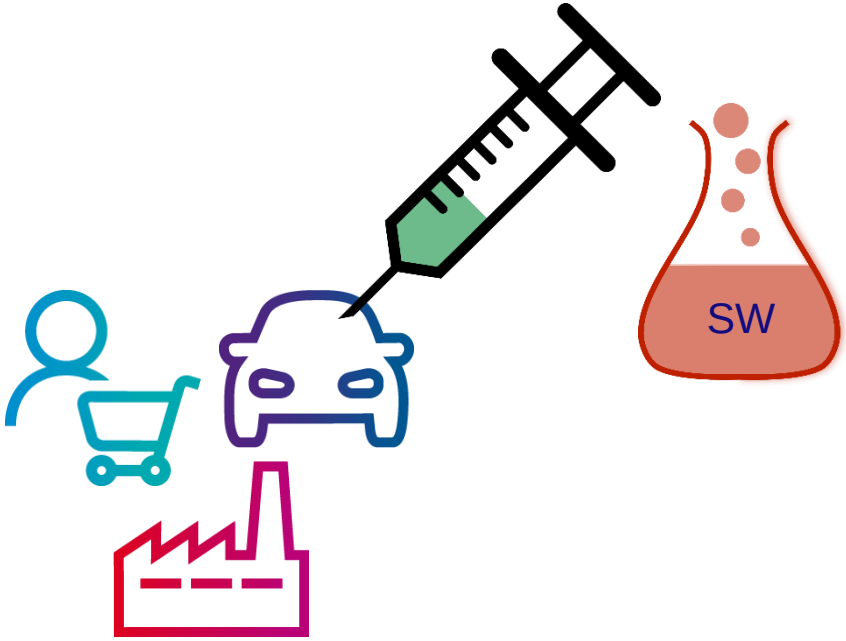
Objective

Bridge the gap

Modeling & Simulation

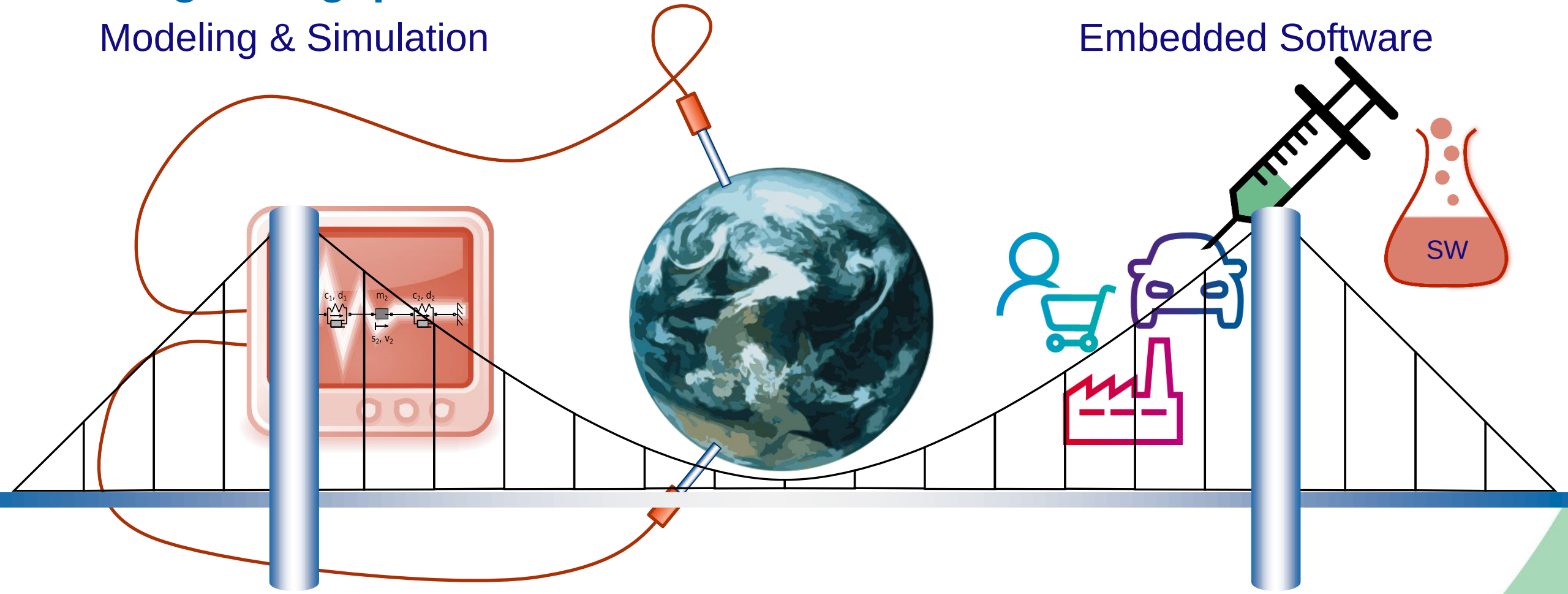


Embedded Software



Objective

Bridge the gap

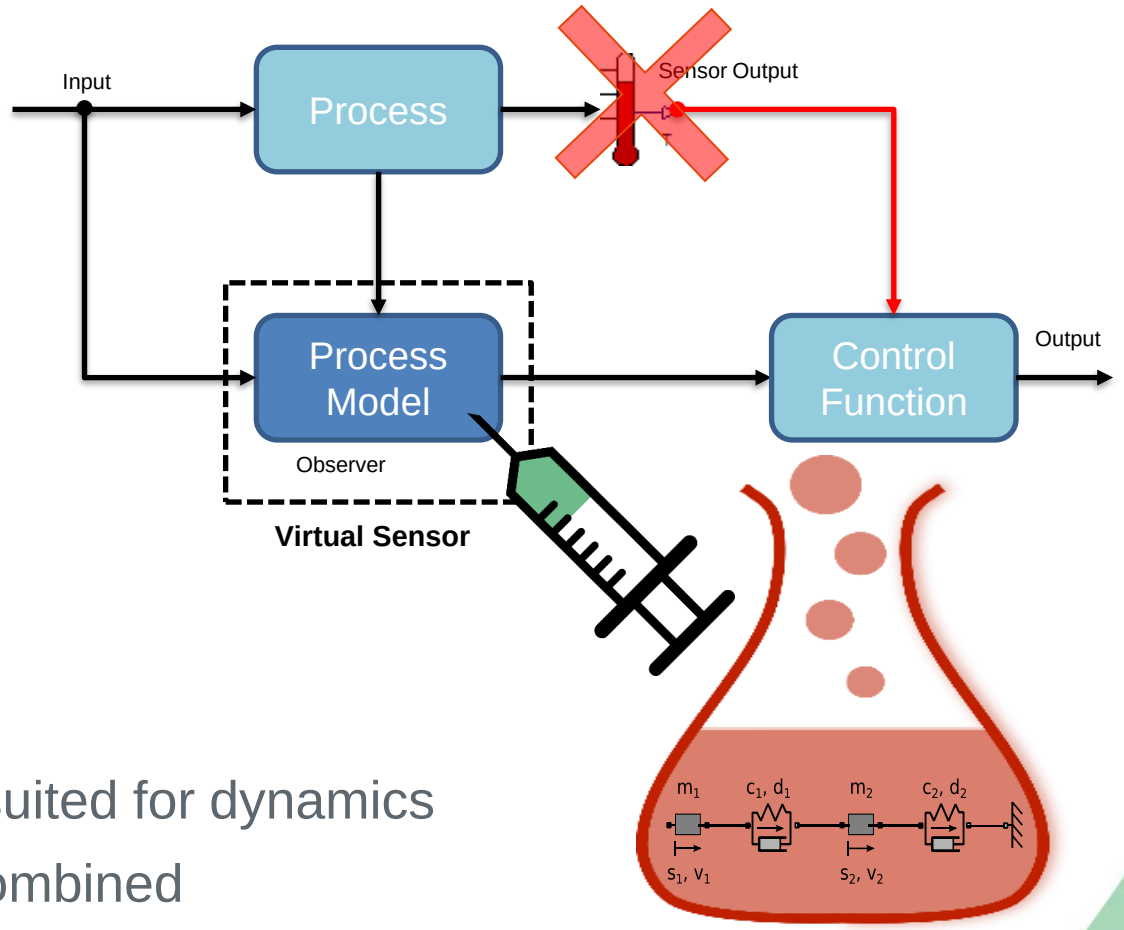


Motivation

Physics-based models for embedded software

Online physical models key technology for advanced engine control software:

- virtual sensors, i.e., observers,
- model-based diagnosis,
- inverse physical models as feed forward part of control structures, and
- model predictive control.

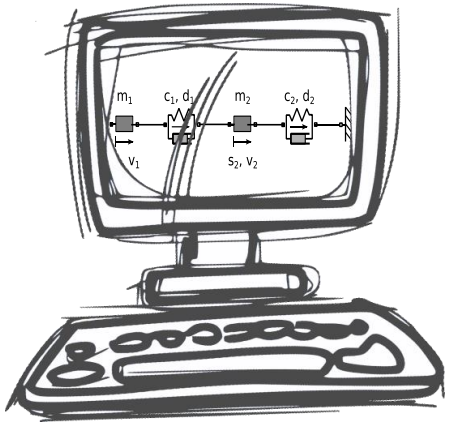


Physical models:

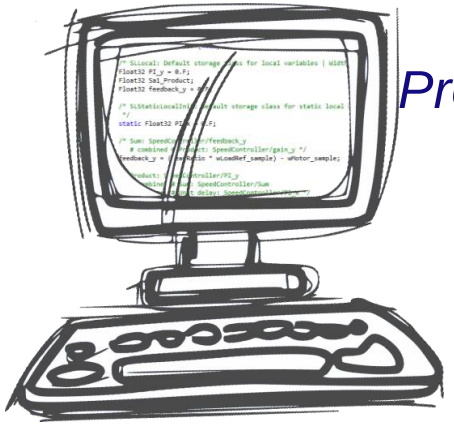
- Typically described by differential equations, best suited for dynamics
- Complementary to data-based modeling, can be combined
- Reduced calibration effort due to physical parameters

Concept

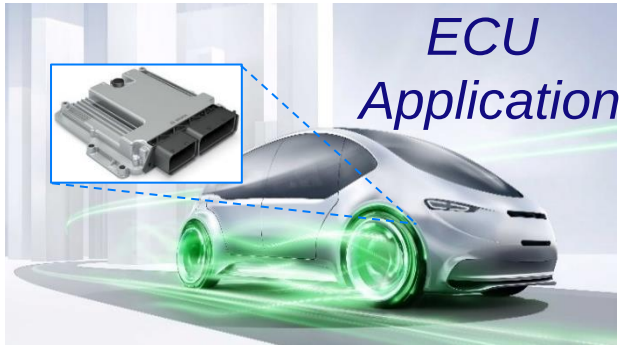
The eFMI workflow



Physics-based Model

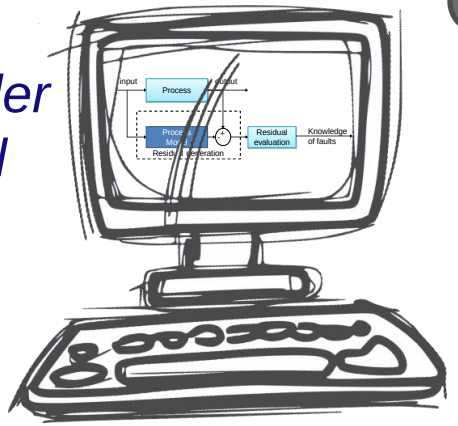


Production Code

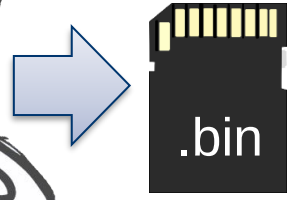


ECU Application

Controller Model

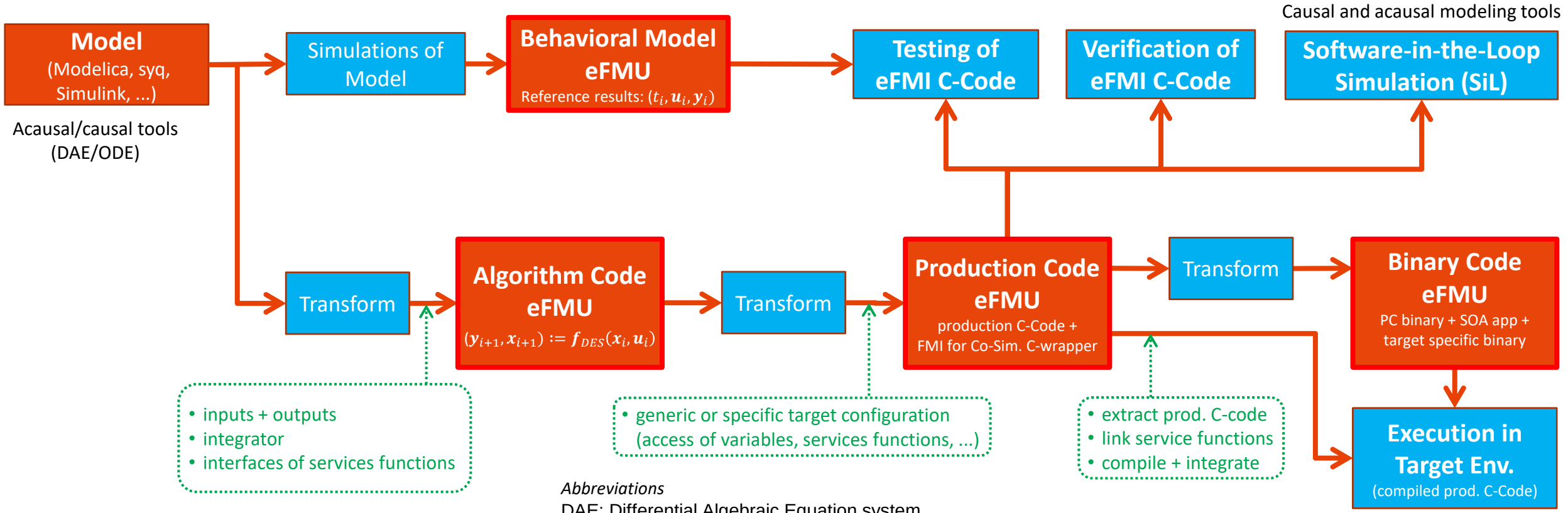


ECU Software



Architecture

The eFMI representations and enabled tool chains



Model

- Physics-based model (e.g. vehicle model)
- Controller, estimator, ...
- Diagnosis system, neural network, ...
- Any combination from above

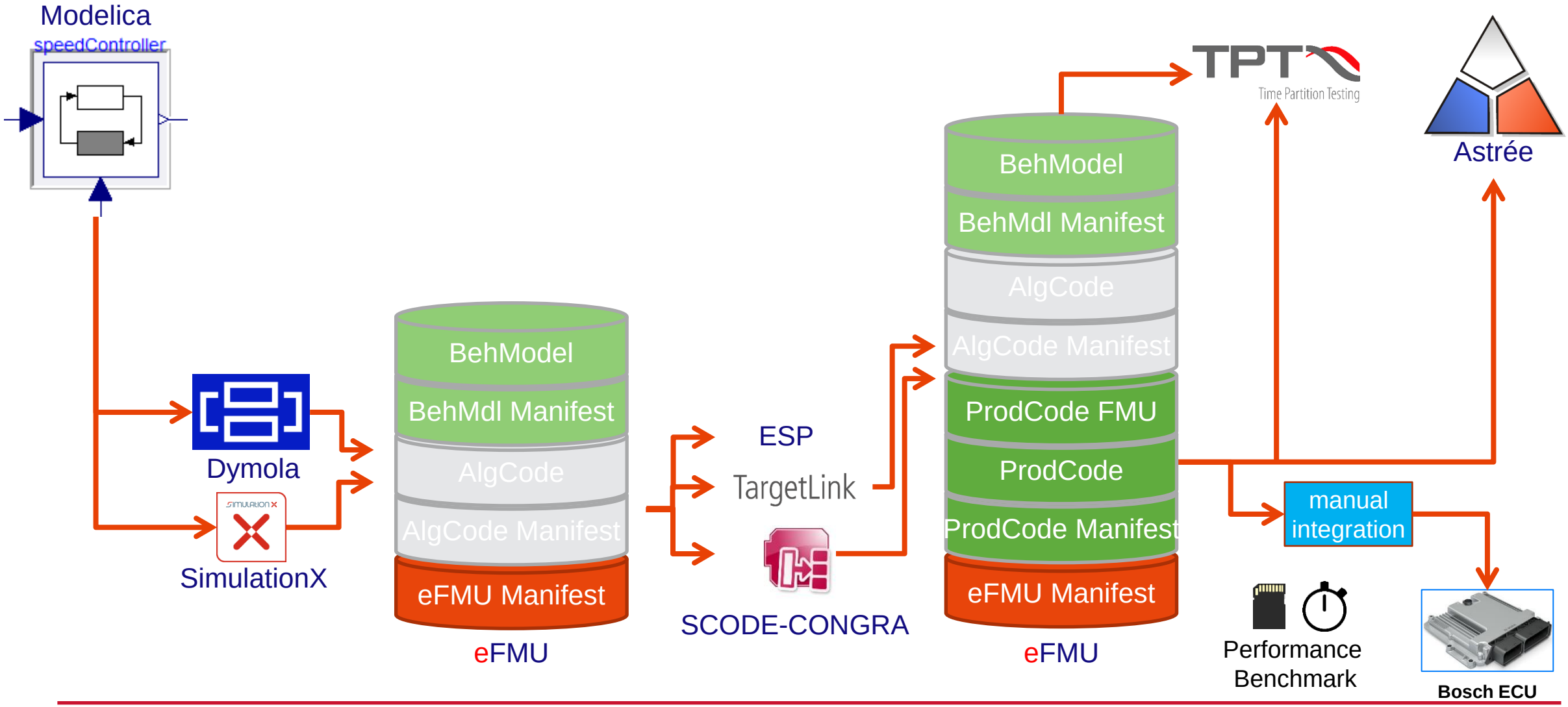
Abbreviations

DAE: Differential Algebraic Equation system
ODE: Ordinary Differential Equation system in state space form
DES: Difference Equation system in State space form
eFMI: Functional Mockup Interface for Embedded systems
eFMU: Functional Mockup Unit for Embedded systems
ECU: Electronic Control Unit

ECU
Realtime-PC
Rapid Prototyping Systems
AUTOSAR
AUTOSAR Adaptive
...

Application

eFMI Tool Chain applied to Speed Controller example



Standardization and Future Work

Schedule

eFMI Standardization

- eFMI specification 1.0.0alpha.3
  - to be published before Feb. 10, 2021
    https://emphysis.github.io/
- Modelica Association Project for eFMI
  - Application will be submitted to the MA before Feb. 10,2021
  - First official release, maintenance and further development.

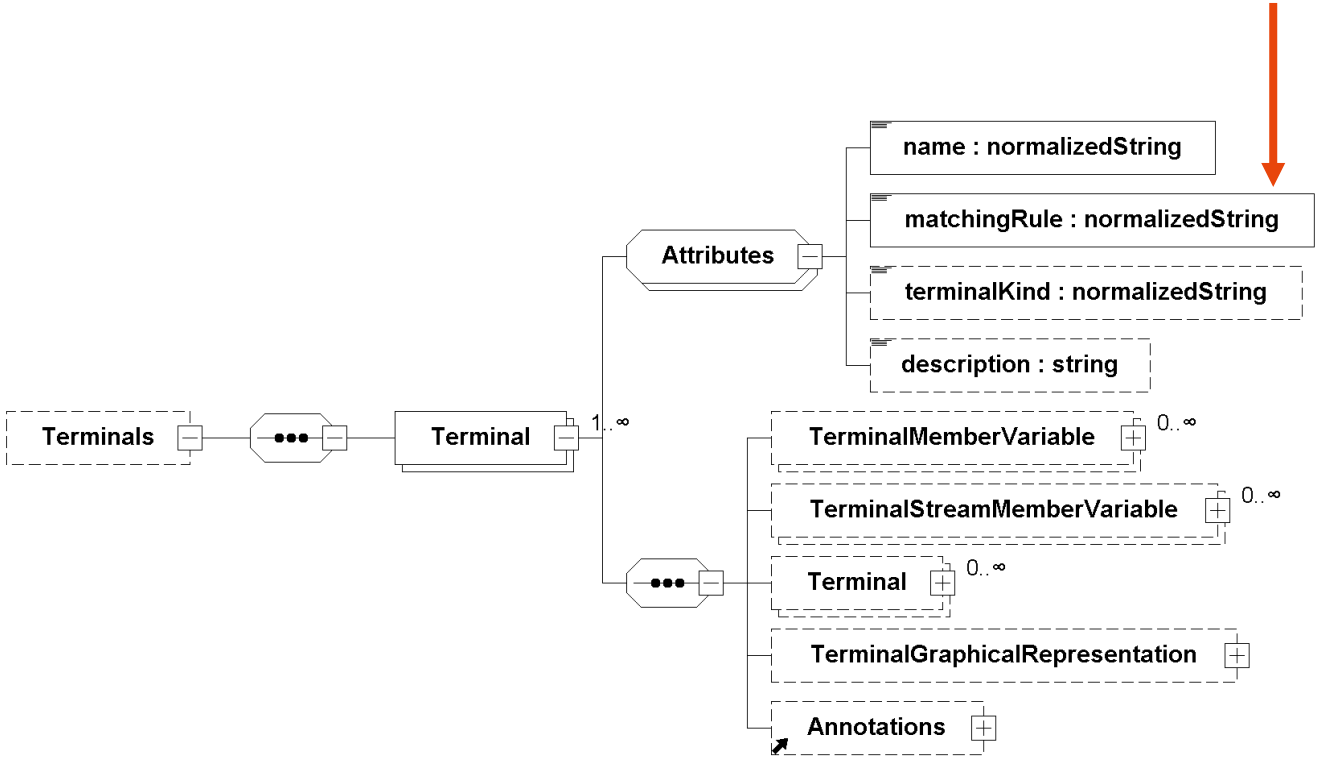
Appendix & Extras

FMI 3.0: New Data Types

FMI 1.0, FMI 2.0	FMI 3.0	Remarks
`fmiReal`	`fmi3Float32`	Discrete and continuous variables
	`fmi3Float64`	States, derivatives, event-indicators
`fmiInteger`	`fmi3Int8, fmi3UInt8`	Discrete variables
	`fmi3Int16, fmi3UInt16`	
	`fmi3Int32, fmi3UInt32`	
	`fmi3Int64, fmi3UInt64`	
`fmiBoolean`	`fmi3Boolean`	`char`
`fmiString`	`fmi3String`	`const char*` ('\0' terminated, UTF-8 encoded)
	`fmi3Binary`	`const char*` (out-of-band length terminated) For large sets of data (sensor outputs, bitmaps, …) Content is specified via `mimeType` in modelDescription.xml
	`fmi3Clock`	Transport information about events

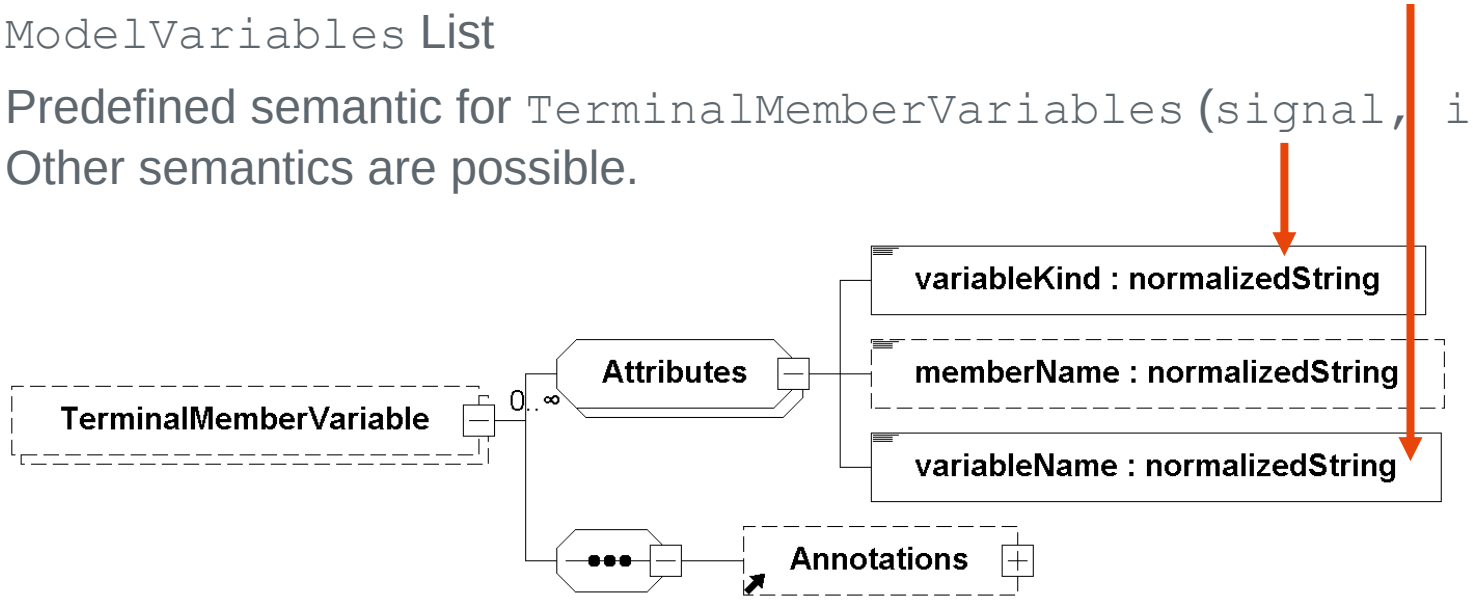
Terminals

- Group inputs and outputs to Terminals, which represent buses or physical connectors
- Predefined matching rules (`plug, bus, sequence`) for the whole Terminal, other rules are possible



Terminals

- `TerminalMemberVariables` reference to variables (inputs, outputs, parameters) of the `ModelVariables` List
- Predefined semantic for `TerminalMemberVariables` (`signal, inFlow, outflow, stream`). Other semantics are possible.



- FMI Terminals **are not acausal**! The **causality (`input, output`) is defined** by the referenced variable!

Clocks

- Clocks synchronize FMUs with the importer and with other FMUs:
  - Clocks carry the information that a specific event happens
  - Clocked variables belong to one clock (a so-called clocked model partition). They change only if this clock is active.
- Clocks allow precise handling of time events (independent from continuous time: `fmi3SetTime()`, or arguments of `fmi3DoStep()`)

- In **Scheduled Execution** Communication Clocks are used:
  - by the importer to identify the specific partition which is to be executed
  - by the FMU to announce, which model partition wants to be scheduled

Concept of Layered Standard

- The layered standard concept allows the specification of standards on top of FMI
- XML element annotations and strings allow additional semantic for variables and terminals
- Extra folder in FMU zip-file allows shipping of additional files at a well-defined place without disturbing compatibility

- Examples:
  - XCP: When packaging virtual electronic control units (vECUs), XCP allows standardized access (see ASAM) to ECU internal variables (in preparation on FMI GitHub)
  - CAN2Signals: Allows grouping and description of FMU inputs and outputs as CAN bus signals (in preparation on FMI GitHub)
  - Including of 3D-Visualization to FMUs which represent multi body simulation models (prototype from ESI ITI and TU Dresden)

Miscellaneous

- Graphical representations for the whole FMU and for Terminals can be defined
- Alias variable names are now specified by a list of alias names for each variable and no longer by a separate variable with the same `valueReference`.
- Dependencies might change at runtime due to variable structure of the model or due to changes of array sizes. Dependencies for (array) variables can now be retrieved at runtime.
- Asynchronous execution of `fmi2DoStep` was removed for simplification. This feature was never used and can be implemented by the importer.
- Improvement and clarification of source code FMUs for better platform independency.

EMPHYSIS Consortium

Acknowledgements

Germany

- Bosch[1,3]
- DLR[2]
- ETAS
- ESI ITI
- AbsInt
- PikeTec
- dSPACE
- EFS

Sweden

- Dassault Systèmes AB[3]
- Volvo Cars
- Modelon
- Linköping University
- SICS East

- **France**
  - Siemens SAS[3]
  - Dassault Systèmes SE
  - Renault
  - CEA
  - University of Grenoble
  - FH Electronics
  - OSE
  - Soben
- **Belgium**
  - Siemens NV[3]
  - Dana
  - University of Antwerp
- **Canada***
  - Maplesoft[3]

- **OEM Advisory Board**
  - BMW
  - Daimler
  - Mazda
  - Volvo

* w/o funding
1) Project Lead
2) Technical Coordination
3) National Coordination