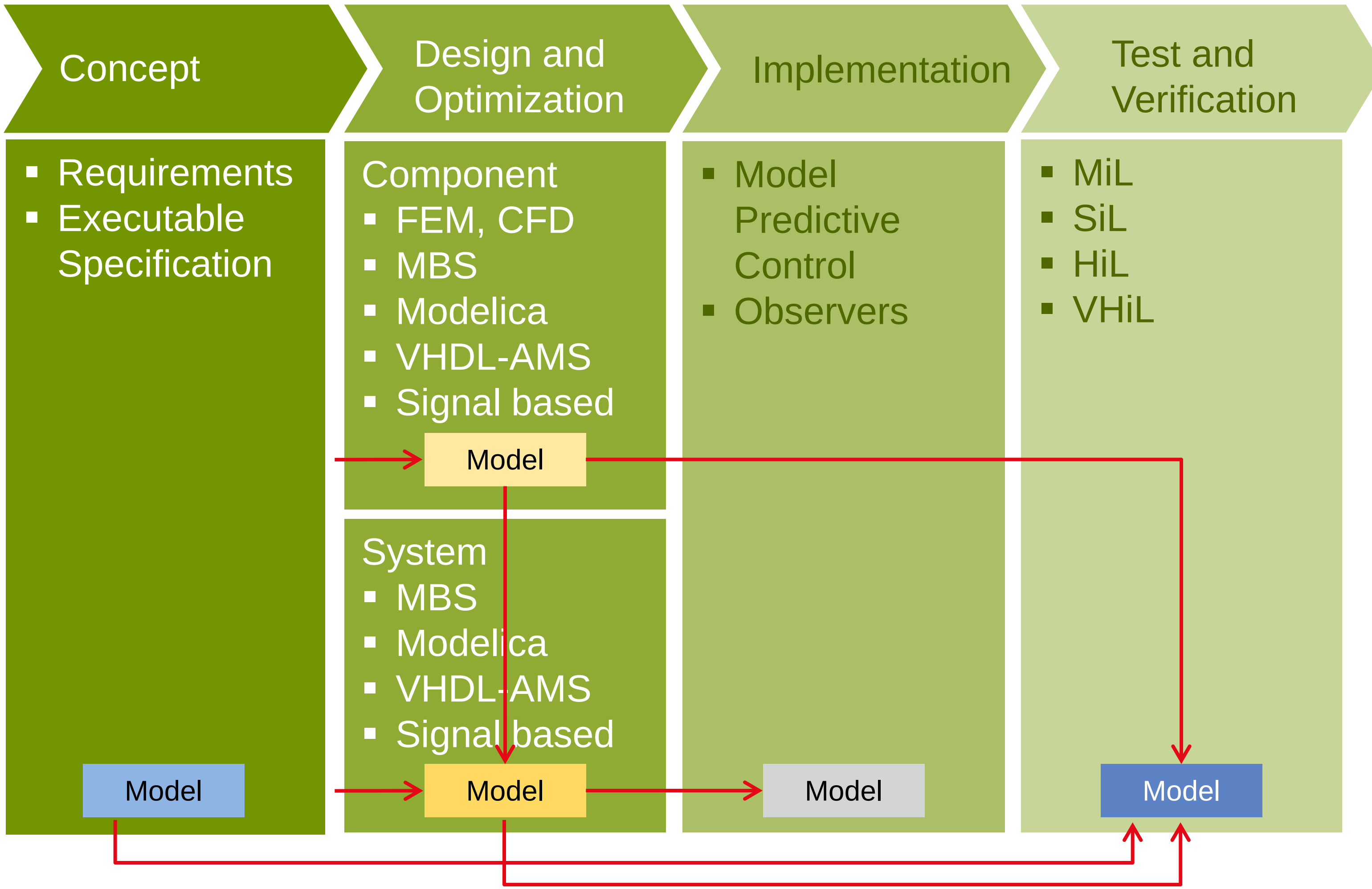
Functional Mockup Interface for Model Exchange and Co-Simulation

T. Blochwitz	ITI, Dresden
M. Otter	DLR, Oberpfaffenhofen
J. Akesson	Modelon, Lund,
M. Arnold	University of Halle
C. Clauß	Fraunhofer IIS EAS, Dresden
H. Elmqvist, H. Olsson	Dassault Systèmes, Lund
M. Friedrich,	Simpack AG, Gilching
A. Junghanns, J. Mauss	QTronic, Berlin
D. Neumerkel	Daimler AG, Stuttgart
A. Viel	LMS Imagine, Roanne

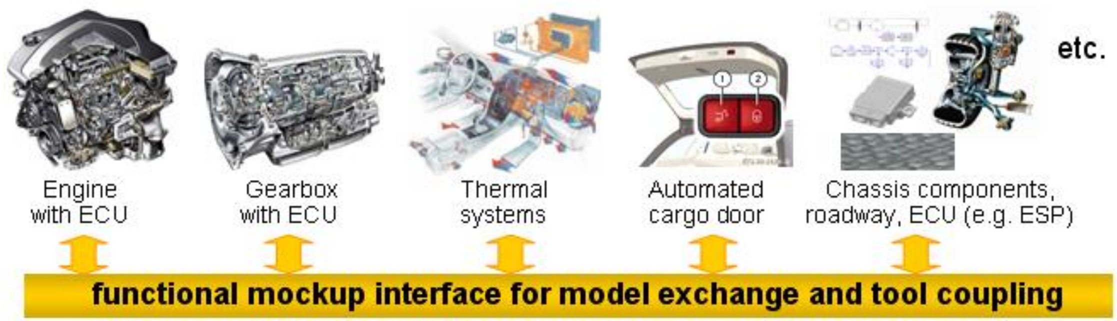
Contents

Motivation

Modeling and Simulation are applied in all stages of system design



Motivation



Challenges for Functional Mockup:

- Different tools and languages are involved
- No standards for model interface and co-simulation available
- Protection of model IP and know-how of supplier

Modelisar project:

- **Functional Mockup Interface for Model Exchange and Co-Simulation**

Functional Mockup Interface

EU project Modelisar (2008 – 2011, 26 Mill. €, 178 my)

- Initiated by Daimler AG, 28 European partners
  - Tool vendors
  - Users
  - Research organizations
- Proof of concept in industrial use cases

After 2011

- Continuation as Modelica Association Project
- Modelica Association changed its bylaws to become an umbrella organization for projects related to model based system design

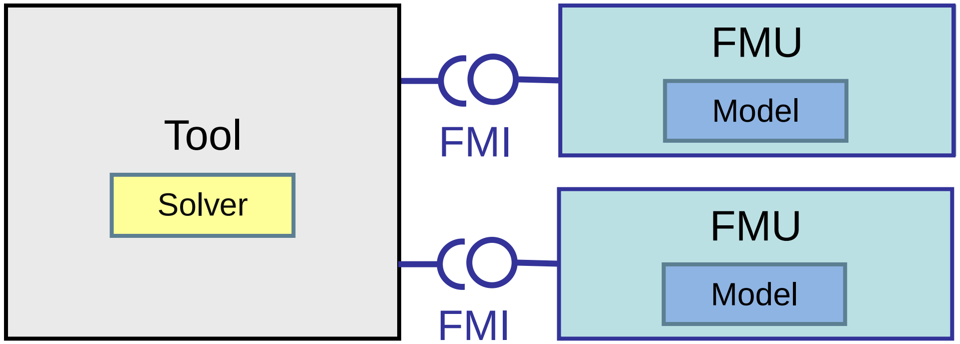
MODELISAR
(ITEA 2 ~ 07006)

Partners

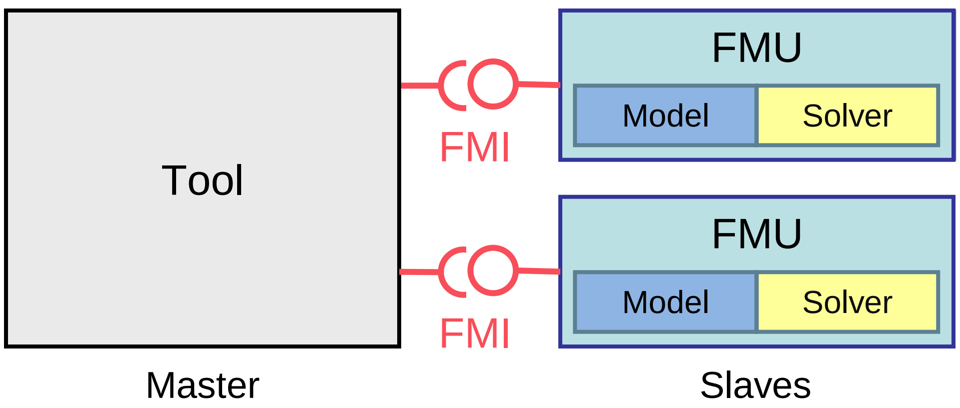
ARMINES
Arsenal Research
ATB
AVL
Berata
Daimler
Dassault Systèmes
David
DLR
Dynasim
Extessy
FhG First, IIS EAS, SCAI
Geensys
Halle University
IFP
Imagine
INSPIRE
SIMPACK AG
ITI
LMS International
QTronic
Schneider Electric
Trialog
Triphase
TWT
Verhaert
Volkswagen
Volvo

FMI – Main Design Idea

- FMI for Model Exchange

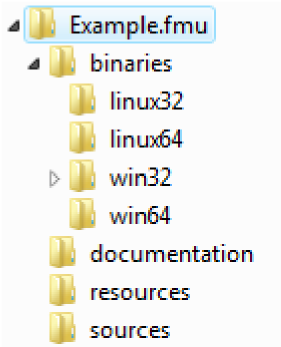


- FMI for Co-Simulation



FMI – Main Design Idea

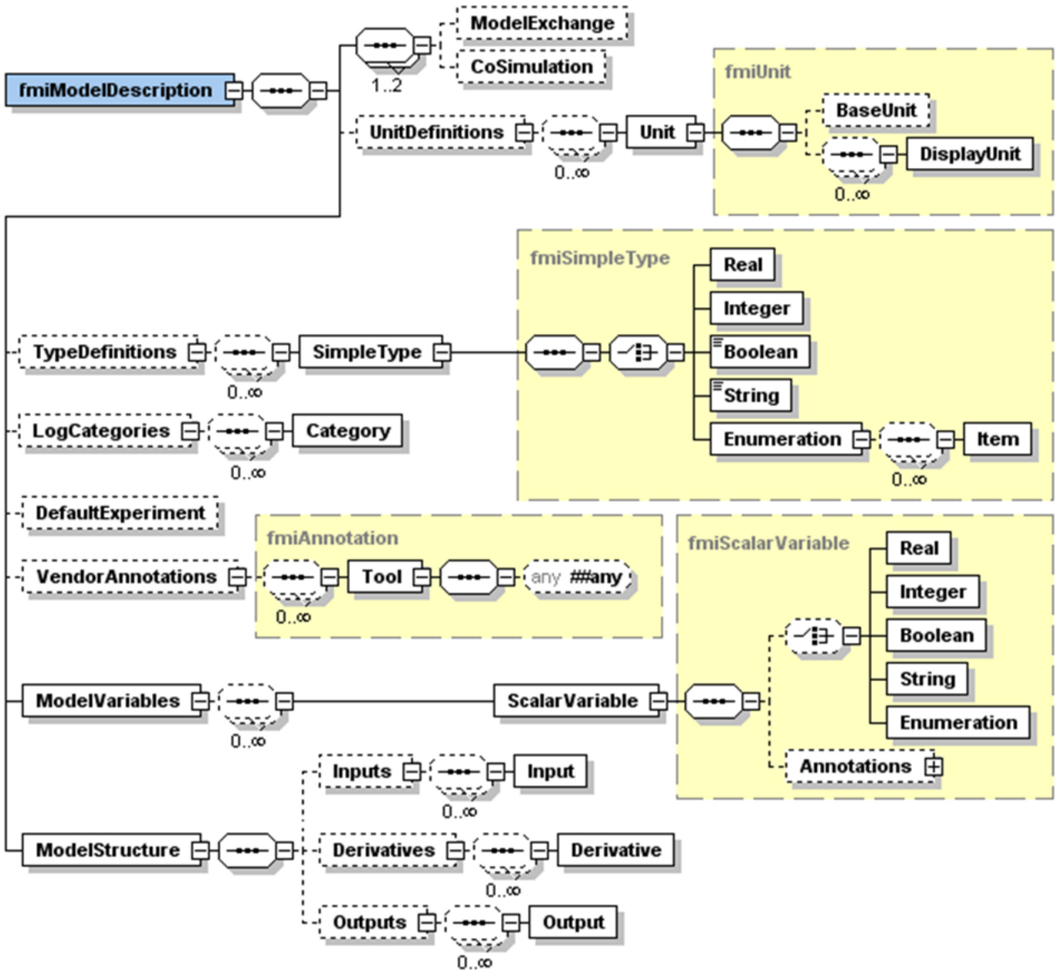
- A component which implements the interface is called a *Functional Mockup Unit (FMU)*
- Separation of:
  - Description of interface data (XML file)
  - Functionality (API in C)
- An FMU is a zipped file (*.fmu) containing:
  - modelDescription.xml
  - Implementation in source and/or binary form
  - Additional data and functionality



- One FMU can contain implementations of both interfaces

XML Model Description

Interface definition is stored in one xml-file:



Implementation and capability flags

Definition of units

Definition of variable types

Variables and their attributes

Dependency information

Example

```xml
<?XML version="1.0" encoding="UTF-8"?>
<fmiModelDescription
  XMLns:xsi="http://www.w3.org/2001/.."
  xsi:noNamespaceSchemaLocation="fmiModel.."
  fmiVersion="2.0"
  modelName="FMU_Coupling.DriveTrain_TorqueAtEnd"
  guid="{a4976b5c-b9f7-432a-9dd3-e80bafaac060}"
  ...>
  <ModelExchange
    modelIdentifier="FMU_0Coupling_..."
    canGetAndSetFMUstate="true"
    providesPartialDerivativesOf_DerivativeFunction_wrt_States="true"
    providesDirectionalDerivatives="true"/>
  <CoSimulation
    modelIdentifier="FMU_0Coupling_..."
    canHandleVariableCommunicationStepSize="true"
    canInterpolateInputs="true"
    .../>
  <UnitDefinitions>
    <Unit name="N.m">
      <BaseUnit kg="1" m="2" s="-2"/> </Unit>
  </UnitDefinitions>
  <TypeDefinitions>
    <SimpleType
      name="Modelica.SIunits.Torque">
      <Real quantity="Torque" unit="N.m"/>
    </SimpleType>
    ...
  </TypeDefinitions>
  <DefaultExperiment startTime="0.0"
    stopTime="1.0" tolerance="0.0001"/>
    ...
```

Example

```xml
...
<ModelVariables>
  <ScalarVariable
    name="torque"
    valueReference="335544320"
    description="Torque in flange"
    causality="output">
    <Real
      declaredType=
       "Modelica.Blocks.Interfaces.RealOutput"
      unit="N.m"/>
   ...
</ModelVariables>
<ModelStructure>
  <Inputs>
    <Input name="phi"/>
    <Input name="w" derivative="1"/>
  </Inputs>
  <Derivatives>
    <Derivative
      name="der(inertia.phi)"
      state="inertia.phi"
      stateDependencies="2"
      inputDependencies=""/>
    <Derivative
      name="der(inertia.w)"
      state="inertia.w"/>
  </Derivatives>
  <Outputs>
    <Output name="torque"
     inputDependencies="1 2"
     inputFactorKinds="fixed fixed"/>
  </Outputs>
</ModelStructure>
</fmiModelDescription>
```

C-Interface

- Instantiation:

```
fmiComponent fmiInstantiateModel(fmiString instanceName, ...)
fmiComponent fmiInstantiateSlave(fmiString instanceName, ...)
```

  - Returns an instance of the FMU. Returned `fmiComponent` is an argument of the other interface functions.
- Functions for initialization, termination, destruction
- Support of real, integer, boolean, and string inputs, outputs, parameters
- `Set` and `Get` functions for each type:

```
fmiStatus fmiSetReal   (fmiComponent c,
                        const fmiValueReference vr[], size_t nvr,
                        const fmiReal value[])
fmiStatus fmiSetInteger(fmiComponent c,
                        const fmiValueReference vr[], size_t nvr,
                        const fmiInteger value[])
```

- Identification by `valueReference`, defined in the XML description file for each variable

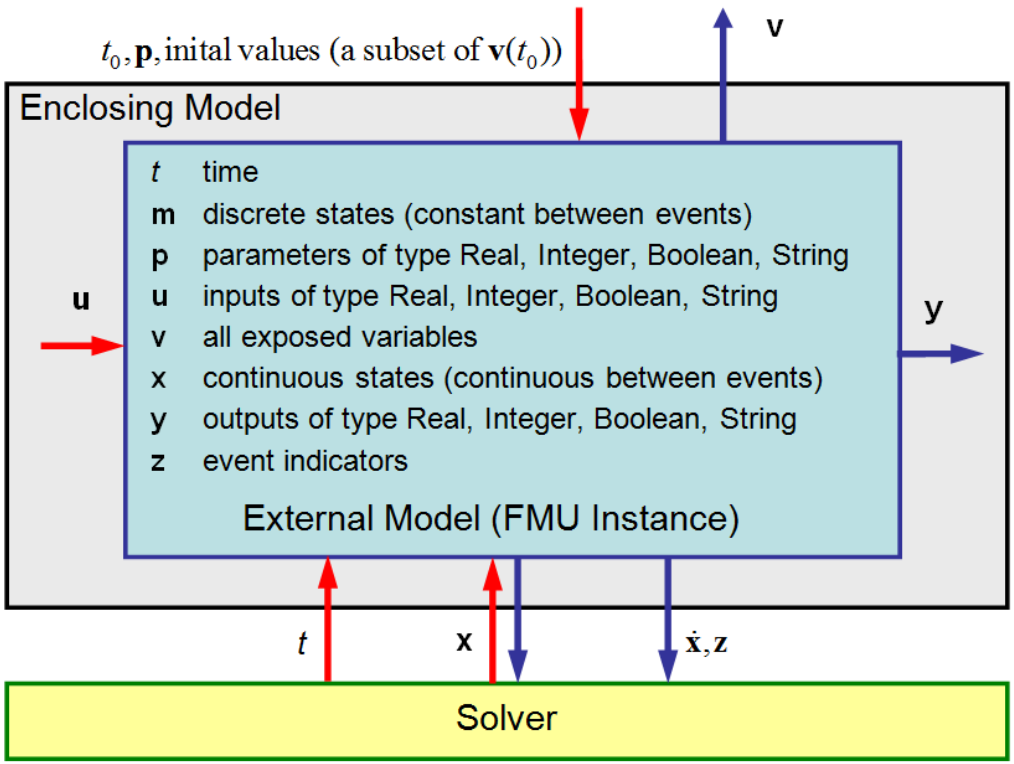
FMI for Model Exchange
Features

- Functionality of state of the art modeling methods can be expressed
- Support of continuous-time and discrete-time systems
- Model is described by differential, algebraic, discrete equations

- Interface for solution of Ordinary Differential Equations (ODE)
- Handling of time, state and step events, event iteration

- Discarding of invalid inputs, state variables
- No explicit function call for computation of model algorithm
  - FMU decides which part is to be computed, when a `fmiGetXXX` function is called
  - Allows for efficient caching algorithms

FMI for Model Exchange
Signals



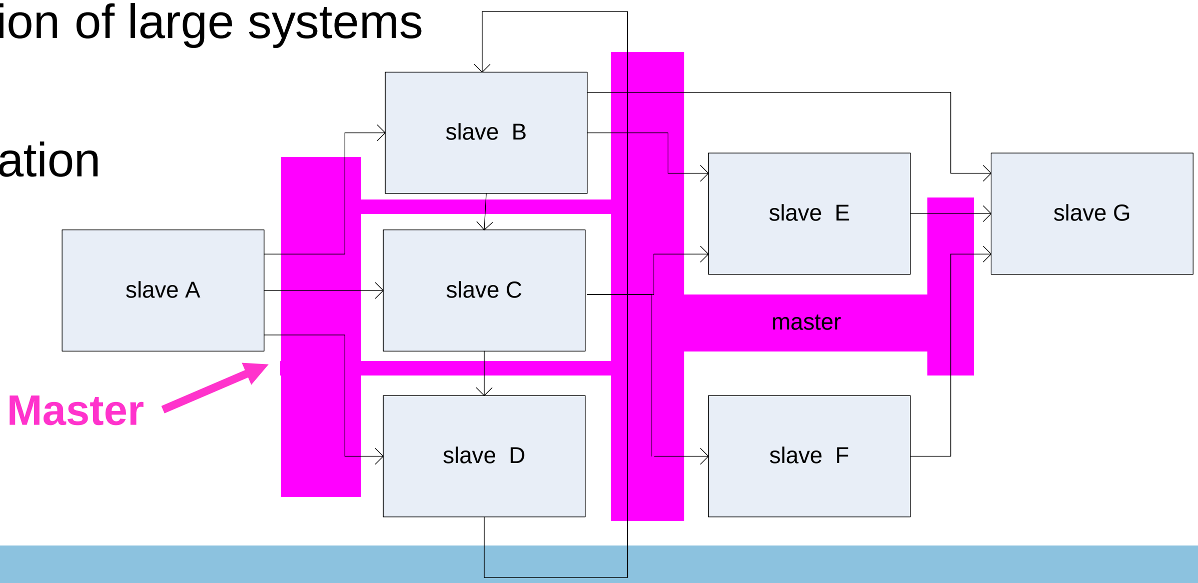
Co-Simulation

Definition:

- Coupling of several simulation tools
- Each tool treats one part of a modular coupled problem
- Data exchange is restricted to discrete communication points
- Subsystems are solved independently between communication points

Motivation

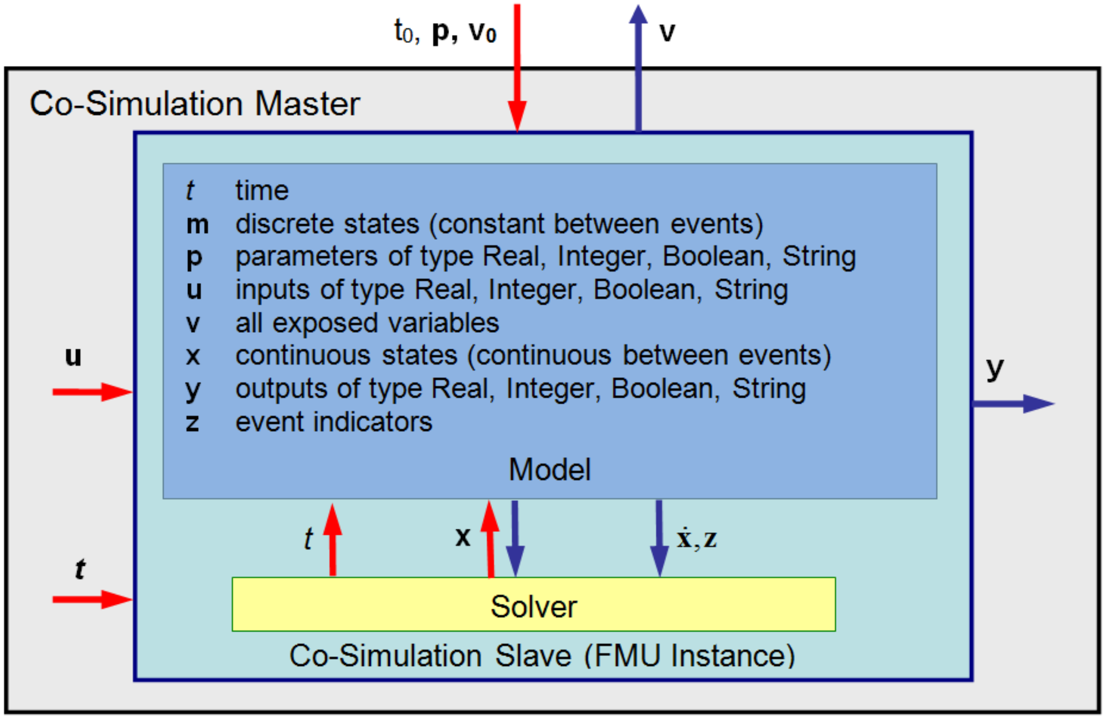
- Simulation of heterogeneous systems
- Partitioning and parallelization of large systems
- Multirate integration
- Hardware-in-the-loop simulation



FMI for Co-Simulation
Features

- State-of-the-Art Co-Simulation:
  - Fixed communication step size
- To improve accuracy and robustness:
  - Optional variable communication step size
  - Optional higher order approximation of inputs and outputs
  - Optional repetition of communication steps
- Capabilities of the slave are contained in the XML-file, for example:
  - `canHandleVariableCommunicationStepSize`
  - `canInterpolateInputs`
  - `canGetAndSetFMUstate`
- Master can decide which coupling algorithm is applicable
- Asynchronous execution (allows parallel execution)

FMI for Co-Simulation
Signals



Additional:

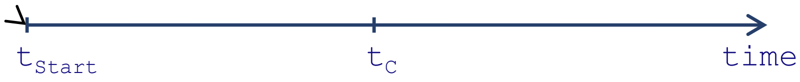
- Status information
- Derivatives of inputs, outputs w.r.t. time for support of higher order approximation

FMI for Model Exchange and Co-Simulation

Sample Code

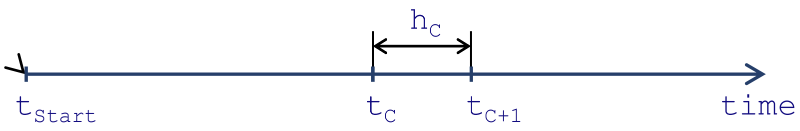
- Model Exchange:
  (One model evaluation)

```
/* Set inputs*/
fmiSetReal(m, id_u1, u1, nu1);
fmiSetTime(m, tC);
fmiSetContinuousStates(m, x, nx);
/* Get results */
fmiGetDerivatives(m, derx, nx);
fmiGetEventIndicators (m, z, nz);
fmiGetReal(m, id_u1, u1, nu1);
```

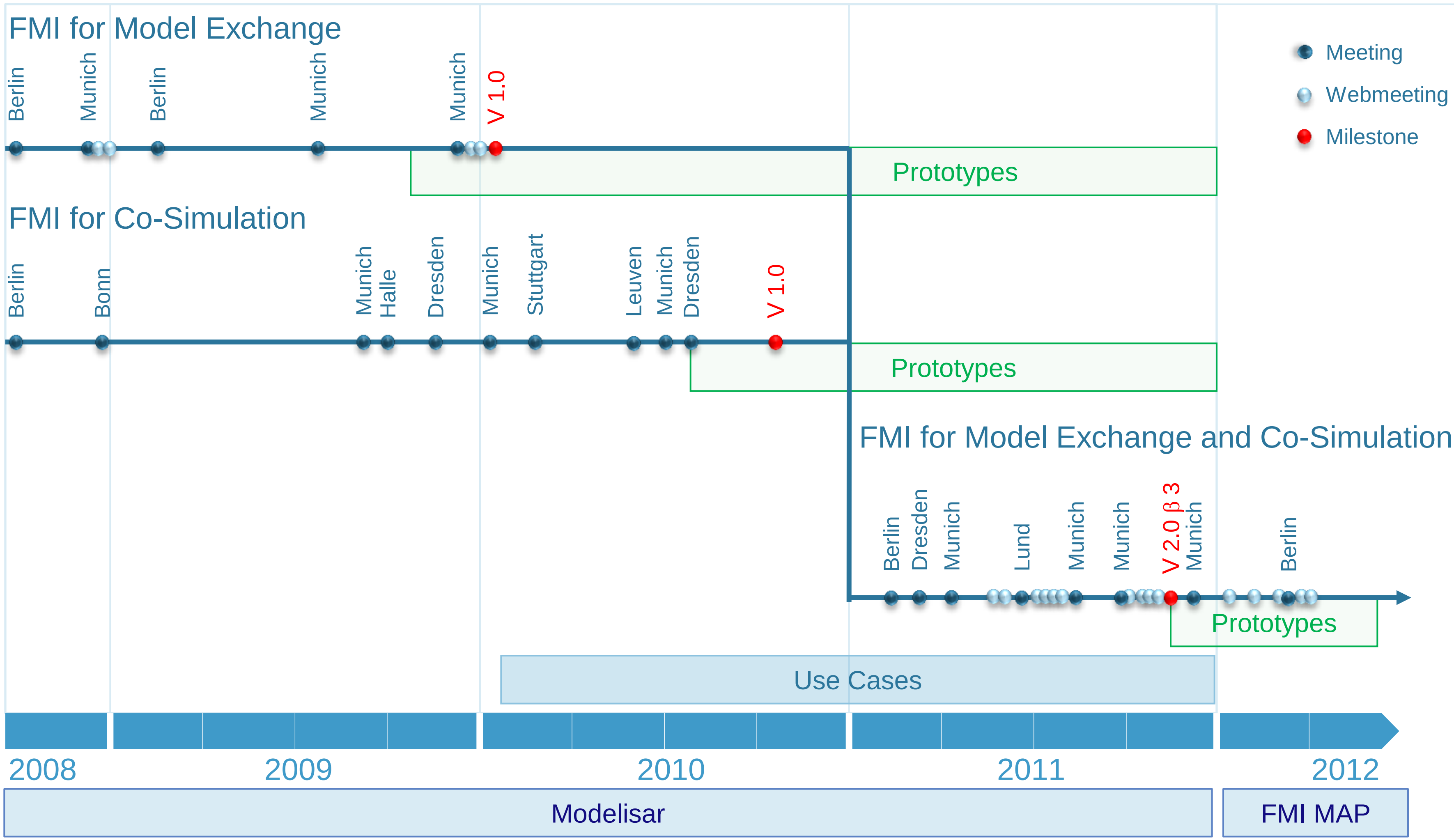


- Co-Simulation:
  (One communication step)

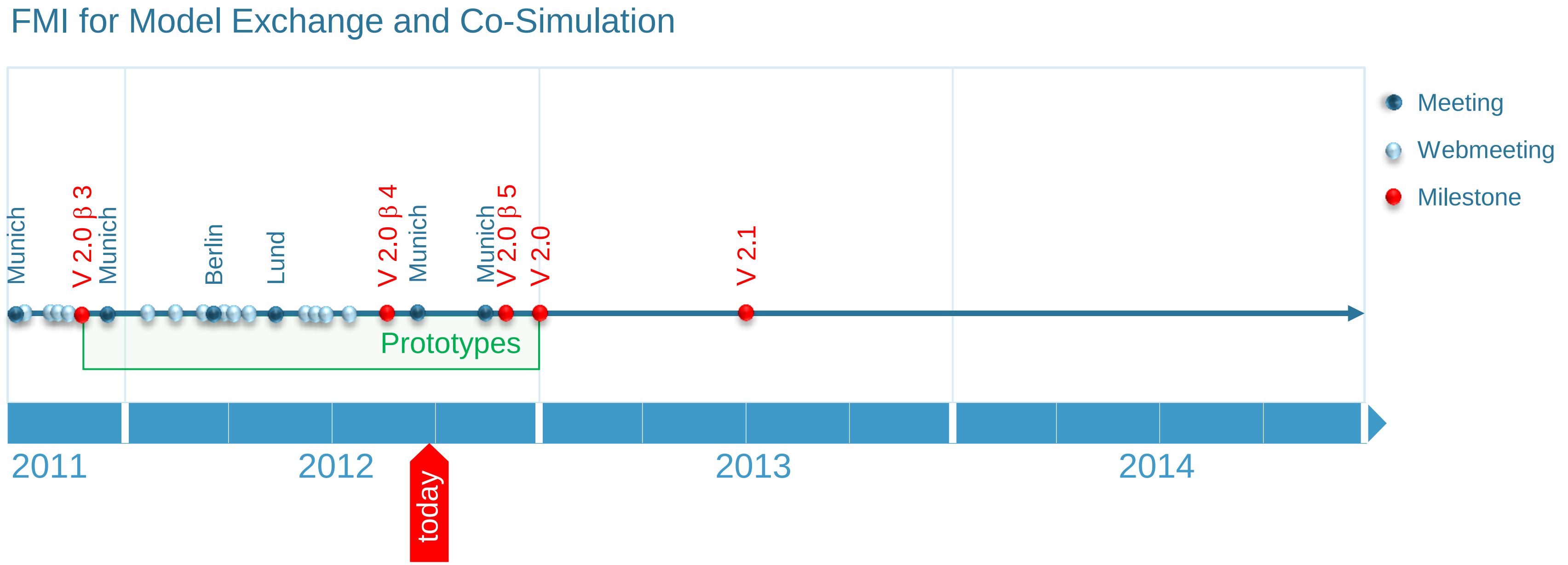
```
/* Set inputs*/
fmiSetReal(s, id_u1, u1, nu1);
/* Do computation*/
fmiDoStep(s, tC, hC, fmiTrue);
/* Get results */
fmiGetReal(s, id_u1, u1, nu1);
```



Development Process



Development Process



FMI 2.0 specification:

- Release December 2012
- Valid for several years
- Backwards-compatible enhancements in minor releases

FMI 2.0
New Features

- Motivation for FMI 2.0
  - Clarification of specification document
  - Ease usability
  - Increase performance for large models

- Unification of Model Exchange and Co-Simulation Standard
  - FMU can contain implementations of both interfaces
  - Distributed and tool based use cases now also for Model Exchange

- Many minor changes
  - Definition of log categories
  - Removement of alias and anti alias variables to ease usage
  - Continuous state variables are named and ordered
  - Improved unit handling

Current status of FMI 2.0

Clarification of specification:

- Instantiation
- Classification of variables
- Calling sequence

Features:

- Tunable parameters
- Improved unit handling
- Save and restore FMU state
- Detailed dependency information (inputs, outputs, derivatives)
- Efficient interface to partial derivatives

Contained in public Beta 4

- Improved handling of time events

Under Discussion

FMI 2.0
Classification of interface variables

`causality`

- `parameter`
- `input`: output of another model
- `output`: input for another model
- `local`: not to be used by other models

`variability`

- `constant`
- `fixed`: constant after initialization
- `tunable`: constant between events
- `discrete`: changes at event instances
- `continuous`

- Combination of `causality` and `variability` allows clear classification of all kinds of variables
- New: distinction between `tunable` and `fixed parameters`
  - Stop simulation, set tunable parameters, resume simulation

FMI 2.0
Save and Restore FMU State

- FMI 1.0: implicite save and restore depending on arguments of `fmiDoStep`
- FMI 2.0: explicite function calls
```
fmiStatus fmiGetFMUstate(fmiComponent c, fmiFMUstate* FMUstate)
fmiStatus fmiSetFMUstate(fmiComponent c, fmiFMUstate  FMUstate)
```

- Iterative co-simulation algorithms
  - Repeat more than one communication step

- Model Predictive Control
  - Simulate some steps starting from the same state with different sets of input values
  - Use the optimal set as control value for the real system

- FMU state can be serialized into a byte vector
  - Usage: start a training simulator from a certain scenario

FMI 2.0
Dependency Information

- FMI 1.0:
  - Only dependencies of outputs on inputs can be indicated

- FMI 2.0:
  - Dependencies of outputs on continuous states
  - Dependencies of derivatives on continuous states and inputs

- Usage:
  - Detection of algebraic loops
  - Definition of sparsity pattern of Jacobian matrices

FMI 2.0
Dependency Information

- Kind of dependency is also defined:
  - `nonlinear`: Jacobian entry is not constant
  - `fixed`: Jacobian entry is constant
  - `discrete`: Jacobian entry may change after events

- Allows optimizations:
  - Generate linear systems of equations for solution of algebraic loops if possible
  - Reduce number of Jacobian computations

FMI 2.0
Directional Derivatives (Jacobian Matrices)

- Jacobians are needed for:
  - Implicit integration methods
  - Solution of systems of equations resulting from algebraic loops
  - Linearization of FMU
  - Extended Kalman filters

- Numerical computation is expensive for large models

- Optional function for providing directional derivatives
  `fmiStatus fmiGetDirectionalDerivative(fmiComponent c,..)`
- Arguments define which derivative(s) w.r.t. which variable(s) are to be retrieved

FMI 2.0
Time Event Handling (under Development)

Requirements:

- Guarantee synchronicity of time events
- Support a subset of the synchronous extensions from Modelica 3.3 (time triggered clocks with constant and variable period)
- Allow backward compatible extensions
- Usable for tools without synchronous features

Main design idea:

- FMU exposes base rates and clocks in the XML model description
- Clock ticking is signaled by `fmiSetClock(..)` before `fmiEventUpdate(..)`
- Discrete variables can be associated with clocks (optional) in XML model description

FMI 2.1
Hierarchical Data, Buses, Physical Connectors (planned)

Requirements:

- Group variables to hierarchical structures, connectors
- Signal based tools must not be excluded
- Keep type information of connectors (e.g. `Modelica.Electrical.Analog.Interfaces.Pin`)
- Add connector type definition for reconstruction of connector type or mapping to existing types

Main design idea:

- Additional “layer” in XML model description
- Mark input/output variables as flow or across quantities
- Causality (input, output) is fixed

Roadmap

2012:

- Finalize time event handling
- October: FMI Meeting
- November: Release of public beta 5
- December: Release of FMI 2.0
- Coordinated prototype implementations by tool vendors

2013:

- Backwards-compatible extensions
- Support of arrays and hierarchical data
- Bus and physical connectors
- Graphical appearance
- …

FMI Support in Tools

fmi-standard.org/tools

- Tool support started immediately after release of FMI 1.0
- **32** tools support FMI, **9** intend to
- Within Modelisar project: **15**

Tools supporting FMI	Model-Exchange		Co-Simulation		Notes
	Export	Import	Slave	Master	
AMESim	available	available	planned		Modelica environment from LMS-Imagine
ASIM	planned		planned		AUTOSAR Builder from Dassault Systèmes
Atego Ace		available		available	Co-simulation environment with AUTOSAR and HIL support
Building Controls Virtual Test Bed				planned	Software environment, based on Ptolemy II, for co-simulation of, and data exchange with, building energy and control systems.
CATIA V6R2013	available	available	planned	planned	Environment for Product Design and Innovation, including systems engineering tools based on Modelica, by Dassault Systèmes
Cybernetica CENIT		available		planned	Industrial product for nonlinear Model Predictive Control (NMPC) from Cybernetica.
Cybernetica ModelFit		available		available	Software for model verification, state and parameter estimation, using logged process data. By Cybernetica.
Control Build	available	available via a specific plug-in	available	available	Environment for IEC 61131-3 control applications from Dassault Systèmes
CosiMate		available		available	Co-simulation Environment from ChiasTek
DSHplus	planned		planned		Fluid power simulation software from FLUIDON
Dymola 2012	available	available	available	planned	Modelica environment from Dassault Systèmes
EnergyPlus			planned	available	Whole building energy simulation program.
TWT Matlab/Simulink FMU Interface			available	available	FMI-compatible plug-and-play interface to Matlab/Simulink, available as an integrated block
FMI Library		available		available	Open source (BSD) C library for integration of FMI technology in custom applications by Modelon.
FMU Trust Centre			available		cryptographic protection and signature of models including their safe PLM storage; secure authentication and authorization for protected (co-)simulation
FMU SDK	available	available	available	available	FMU Software Development Kit from QTronic
IPG CarMaker	planned		planned		via Modeling and Co-Simulation environment by Modelon
JFMI		available		available	A Java Wrapper for the Functional Mock-up Interface, based on FMU SDK.
JModelica.org	available	available	planned	planned	Open source Modelica environment from Modelon
MapleSim	available	planned	planned	planned	Modelica-based modeling and simulation tool from Maplesoft
MATLAB		available		available	via FMI Toolbox from Modelon
Microsoft Excel				planned 2012	via FMI Add-in to Microsoft Excel by Modelon. Offers support for batch simulation of FMUs.
MWorks 2.5	available	planned	planned	planned	Modelica environment from Suzhou Tongyuan
NI VeriStand				planned	Real-Time Testing and Simulation Software from National Instruments
FMI add on for NI VeriStand	available	available		available	manages simulations with 'FMI for co simulation V1.0' available from DOFware
NI LabVIEW		planned			Graphical programming environment for measurement, test, and control systems from National Instruments
OpenModelica	available	available			Open source Modelica environment from OSMC
OPTIMICA Studio	available	planned	planned	planned	Modelica environment from Modelon
Ptolemy II				planned	Software environment for design and analysis of heterogeneous systems.
Python		available			via the open source package PyFMI from Modelon. Also available as part of the JModelica.org platform
Silver 2.3.1		available		available	Virtual integration platform for Software in the Loop from QTronic
SIMPACK 9	planned (2012)	available	planned (2011)	available	High end multi-body simulation software from SIMPACK AG
SimulationX 3.4	available	available	available	available	Modelica environment from ITI
Simulink	available				via Dymola 2012 using Real-Time Workshop
Simulink	available				via @Source
Simulink		available		available	via FMI Toolbox from Modelon
TISC		available		available	Co-simulation environment from TLK-Thermo
TWT Co-Simulation Framework			available	available	Communication layer tool to flexibly plug together models for performing a co-simulation; front-end for set-up, monitoring and post-processing included
Vertex	planned				Modelica environment from deltatheta
Virtual.Lab Motion	planned	available	available	available	Virtual.Lab Motion is a high end multi body software from LMS International
xMod		available		available	Heterogeneous model integration environment & virtual instrumentation and experimentation laboratory from IFP.

FMI Support in Tools

- Authoring Tools: **12**
- Integration Tools: **20**
  (Co-Simulation master, HiL, optimization, control, analyses)
- Software Development Kits: **3**
  (C, Python, Java)

Tools supporting FMI	Model-Exchange		Co-Simulation		Notes
	Export	Import	Slave	Master	
AMESim	available	available	planned		Modelica environment from LMS-Imagine
ASIM	planned		planned		AUTOSAR Builder from Dassault Systèmes
Atego Ace		available		available	Co-simulation environment with AUTOSAR and HIL support
Building Controls Virtual Test Bed				planned	Software environment, based on Ptolemy II, for co-simulation of, and data exchange with, building energy and control systems.
CATIA V6R2013	available	available	planned	planned	Environment for Product Design and Innovation, including systems engineering tools based on Modelica, by Dassault Systèmes
Cybernetica CENIT		available		planned	Industrial product for nonlinear Model Predictive Control (NMPC) from Cybernetica.
Cybernetica ModelFit		available		available	Software for model verification, state and parameter estimation, using logged process data. By Cybernetica.
Control Build	available	available via a specific plug-in	available	available	Environment for IEC 61131-3 control applications from Dassault Systèmes
CosiMate		available		available	Co-simulation Environment from ChiasTek
DSHplus	planned		planned		Fluid power simulation software from FLUIDON
Dymola 2012	available	available	available	planned	Modelica environment from Dassault Systèmes
EnergyPlus			planned	available	Whole building energy simulation program.
TWT Matlab/Simulink FMU Interface			available	available	FMI-compatible plug-and-play interface to Matlab/Simulink, available as an integrated block
FMI Library		available		available	Open source (BSD) C library for integration of FMI technology in custom applications by Modelon.
FMU Trust Centre			available		cryptographic protection and signature of models including their safe PLM storage; secure authentication and authorization for protected (co-)simulation
FMU SDK	available	available	available	available	FMU Software Development Kit from QTronic
IPG CarMaker	planned		planned		via Modeling and Co-Simulation environment by Modelon
JFMI		available		available	A Java Wrapper for the Functional Mock-up Interface, based on FMU SDK.
JModelica.org	available	available	planned	planned	Open source Modelica environment from Modelon
MapleSim	available	planned	planned	planned	Modelica-based modeling and simulation tool from Maplesoft
MATLAB		available		available	via FMI Toolbox from Modelon
Microsoft Excel				planned 2012	via FMI Add-in to Microsoft Excel by Modelon. Offers support for batch simulation of FMUs.
MWorks 2.5	available	planned	planned	planned	Modelica environment from Suzhou Tongyuan
NI VeriStand				planned	Real-Time Testing and Simulation Software from National Instruments
FMI add on for NI VeriStand		available		available	manages simulations with 'FMI for co simulation V1.0' available from DOFware
NI LabVIEW		planned			Graphical programming environment for measurement, test, and control systems from National Instruments
OpenModelica	available	available			Open source Modelica environment from OSMC
OPTIMICA Studio	available	planned	planned	planned	Modelica environment from Modelon
Ptolemy II				planned	Software environment for design and analysis of heterogeneous systems.
Python		available			via the open source package PyFMI from Modelon. Also available as part of the JModelica.org platform
Silver 2.3.1		available		available	Virtual integration platform for Software in the Loop from QTronic
SIMPACK 9	planned (2012)	available	planned (2011)	available	High end multi-body simulation software from SIMPACK AG
SimulationX 3.4	available	available	available	available	Modelica environment from ITI
Simulink	available				via Dymola 2012 using Real-Time Workshop
Simulink	available				via @Source
Simulink		available		available	via FMI Toolbox from Modelon
TISC		available		available	Co-simulation environment from TLK-Thermo
TWT Co-Simulation Framework			available	available	Communication layer tool to flexibly plug together models for performing a co-simulation; front-end for set-up, monitoring and post -processing included
Vertex	planned				Modelica environment from deltatheta
Virtual.Lab Motion	planned	available	available	available	Virtual.Lab Motion is a high end multi body software from LMS International
xMod		available		available	Heterogeneous model integration environment & virtual instrumentation and experimentation laboratory from IFP.

Quality of FMI Implementations

FMI Compliance Checker

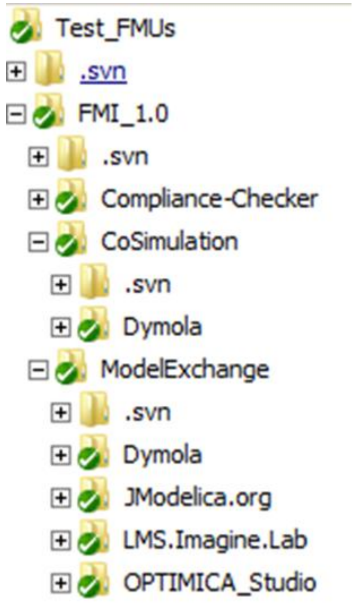
- Open source implementation under contract of MA
- Checks XML model description
- Simulates single FMUs for Model Exchange and Co-Simulation
- https://svn.fmi-standard.org/fmi/branches/public/Test_FMUs/FMI_1.0/Compliance-Checker/

Repository of FMUs, generated by different tools

- https://svn.fmi-standard.org/fmi/branches/public/Test_FMUs

Public Error Tracking System

- https://trac.fmi-standard.org/



Applications outside of Automotive

Power plant simulation and control

- Siemens, ABB, EDF
- EU Project MODRIO (19 Mill. €, 150 man-years, 2012 – 2015)

Building simulation

- Situation is similar to automotive industry:
  - Heterogeneous systems (building, heating, air conditioning, …)
  - Components of different nature and from several suppliers

Research

- Co-Simulation master algorithms
- Model based control

FMI Modelica Association Project (MAP)

General conditions

- FMI project members need not to be Modelica Association (MA) members
- Project results are owned by the MA
- Project results are freely usable under copyleft license
- Meetings are open to the public

FMI Steering Commitee

- Defines FMI policy, strategy, feature roadmap, releases
- Voting rights

FMI Advisory Board

- Contribute to FMI design
- Access to FMI infrastructure (svn, trac, meeting minutes)

FMI Project Rules
How to participate

Steering Commitee

- Prove active FMI support by participation at 2 meetings in the last 24 months
- Support FMI or part of it in a commercial or open source tool, and/or active FMI usage in industrial projects
- Be accepted by Steering Commitee with qualified majority

Advisory Board

- Prove active FMI support by participation at 2 meetings in the last 24 months

Guests

- Send e-mail to contact@fmi-standard.org for registration in mailing list

FMI MAP Members

Steering Committee

- Atego, Daimler, Dassault Systèmes, IFP EN, ITI, LMS, Modelon, QTronic, Siemens, SIMPACK

Advisory Board

- Armines, DLR, Fraunhofer (IIS/EAS, First, SCAI), Open Modelica Consortium, TWT, University of Halle

Guests

- Altair Engineering, Berkeley University, Bosch, ETAS, Equa Simulation, IBM Research

Conclusions

FMI for Model Exchange and Co-Simulation is an established standard

- 32 tools currently support FMI 1.0, 9 intend to
- Is used in industrial and research applications
- Is maintained as Modelica Association Project

FMI project is open for non Modelica tool vendors and organizations

FMI 2.0 improves:

- Compatibility of implementations (clarified specification)
- Usability (tunable parameters, unit handling)
- Efficiency and robustness for large models (dependency information, directional derivatives)

Outlook

FMI 2.0 Release planned for December 2012

Current tasks:

- Precise handling of time events for periodic and aperiodic sampled data systems

Ideas for FMI 2.1

- Arrays, hierarchical data, buses, physical ports
- Graphical appearance, connector placement