Presented at the 15th MODPROD Workshop, 3-4 February 2021,
Linköping, Sweden.
www.modprod.org

fmi Functional Mock-Up Interface

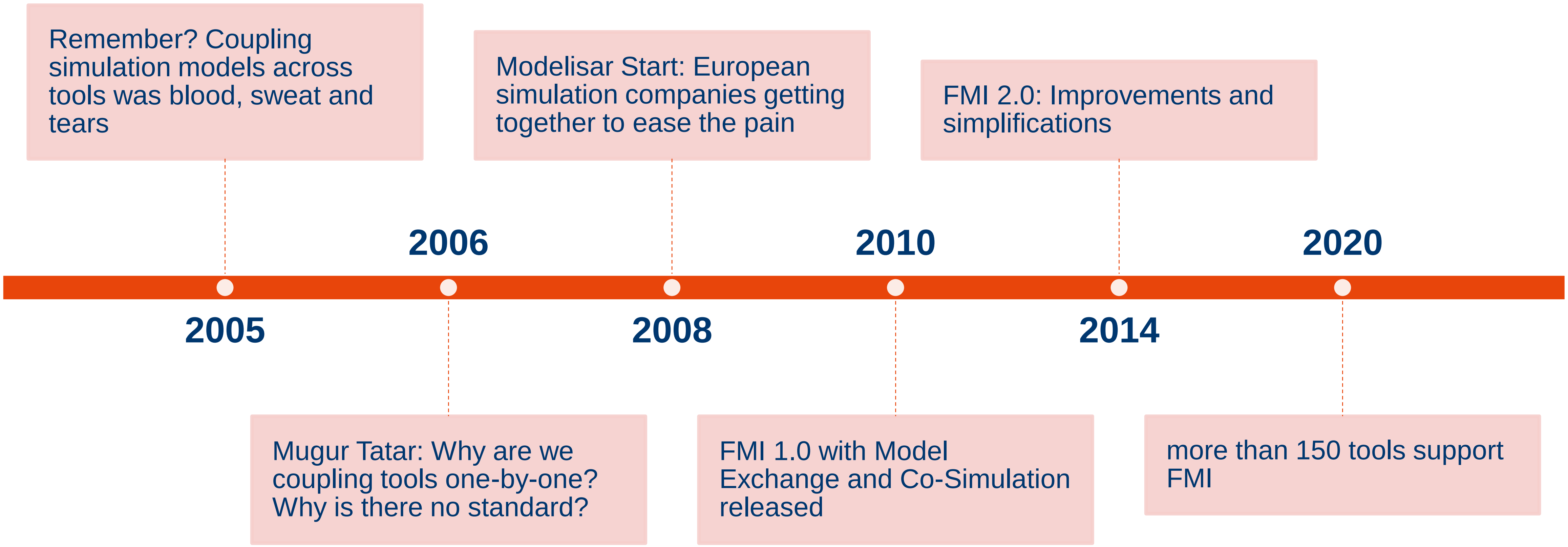
FMI – Current Challenges, Trends and Developments

Andreas Junghanns
Synopsys
FMI Project Leader

Torsten Blochwitz
ESI ITI
FMI Deputy Project Leader

Modelica Association

FMI: Motivation and History



2005

Remember? Coupling simulation models across tools was blood, sweat and tears

2006

Mugur Tatar: Why are we coupling tools one-by-one? Why is there no standard?

2008

Modelisar Start: European simulation companies getting together to ease the pain

2010

FMI 1.0 with Model Exchange and Co-Simulation released

2014

FMI 2.0: Improvements and simplifications

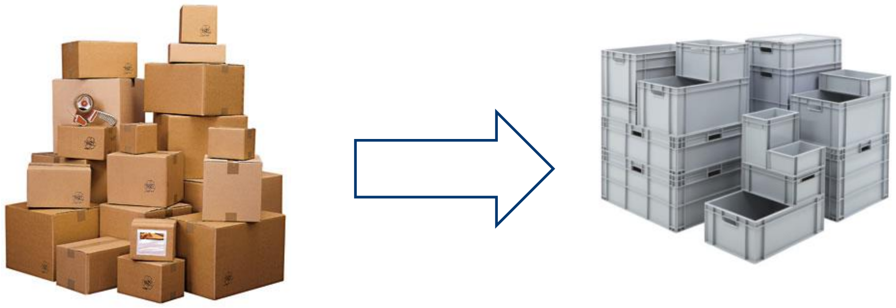
2020

more than 150 tools support FMI

FMI: Simpler „Plumbing“ for Simulation

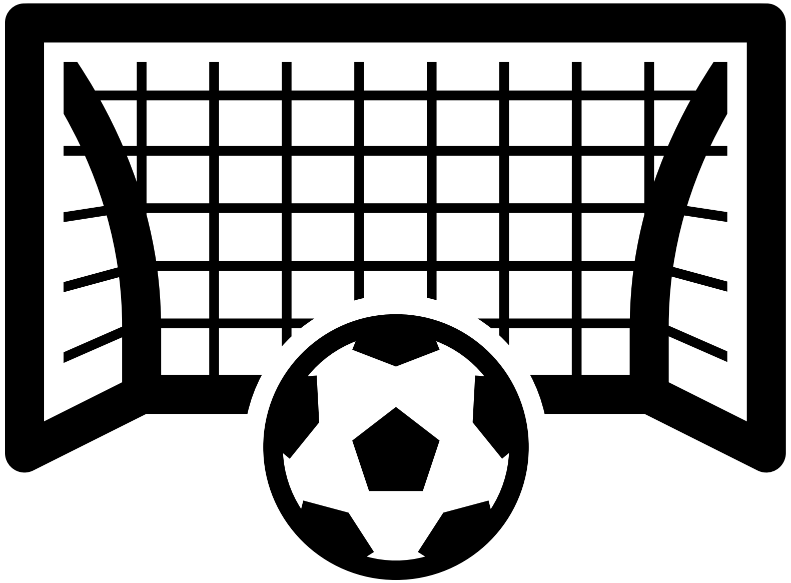
- FMI for Model Exchange:
  How to connect systems of equations (ODEs)
- FMI for Co-Simulation:
  How to connect “any” model or tool

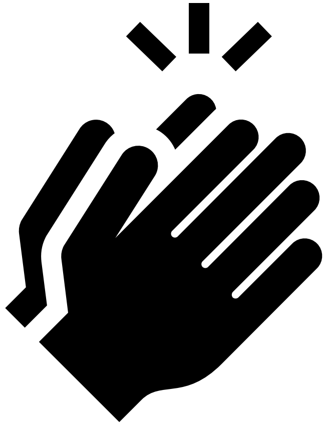
- Decouple Know-How between producers and users of FMUs
- Massive Re-use of modelling investment
- Many new use-cases are now viable
- 150+ tools now support FMI:
  See: fmi-standard.org/tools



https://fmi-standard.org/

FMI: The End?





Mission accomplished?

FMI: Motivation 2020

150+ tools supporting FMI:

- Many more users
- Many more use cases:
  - More cyber physical systems
  - Complex controller code
  - Complex communication
  - Non-numerical values
- Scaling simulations
  - (Signal) handling is getting difficult
- Mostly FMI for Co-Simulation:
  - Causing numerical issues

https://fmi-standard.org/

FMI: Motivation 2020

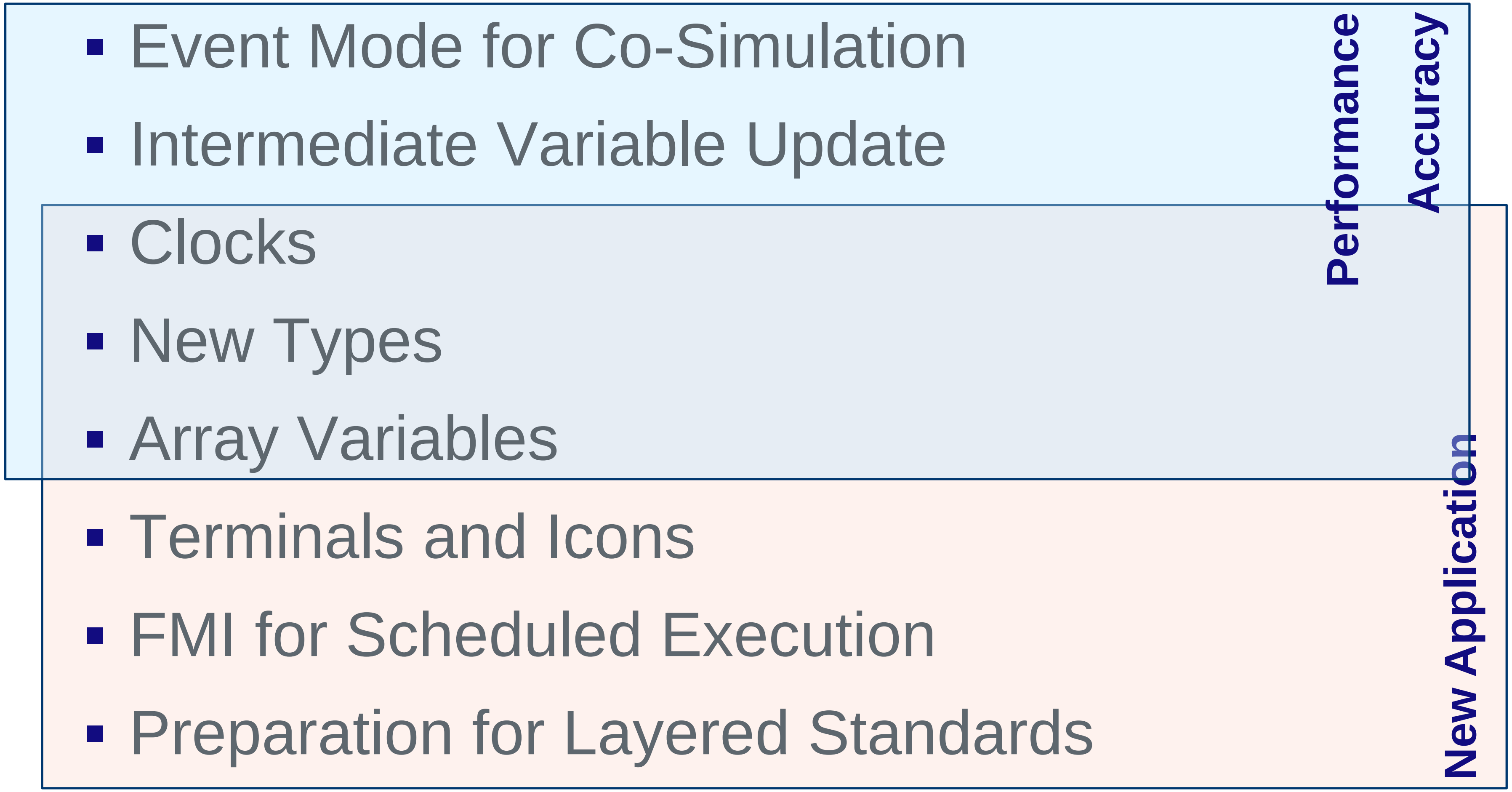
150+ tools supporting FMI:

- Many more users
- Many more use cases:
  - More cyber physical systems
  - Complex controller code
  - Complex communication
  - Non-numerical values
- Scaling simulations
  - (Signal) handling is getting difficult
- Mostly FMI for Co-Simulation:
  - Causing numerical issues

vs.

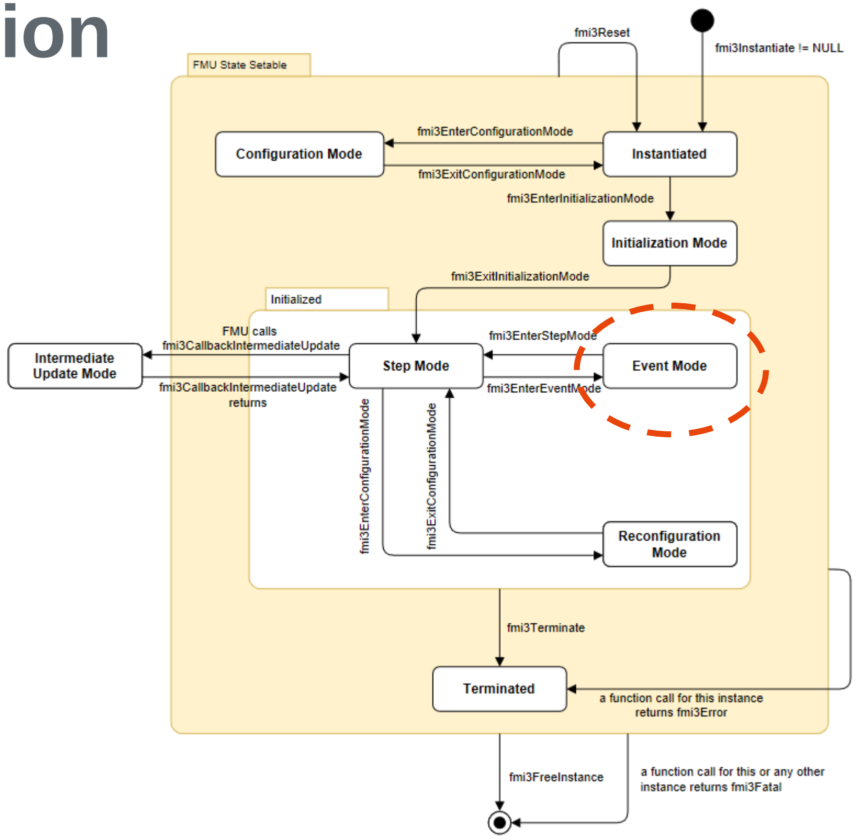
https://fmi-standard.org/

FMI 3.0: Main Improvements



- Event Mode for Co-Simulation
- Intermediate Variable Update
- Clocks
- New Types
- Array Variables
- Terminals and Icons
- FMI for Scheduled Execution
- Preparation for Layered Standards

FMI 3.0: Event Mode also for Co-Simulation


fmi3Reset
fmi3Instantiate != NULL
FMU State Setable
fmi3EnterConfigurationMode
Configuration Mode
Instantiated
fmi3ExitConfigurationMode
fmi3EnterInitializationMode
Initialization Mode
fmi3ExitInitializationMode
Initialized
FMU calls
fmi3CallbackIntermediateUpdate
Intermediate Update Mode
Step Mode
fmi3EnterStepMode
Event Mode
fmi3CallbackIntermediateUpdate returns
fmi3EnterEventMode
fmi3EnterConfigurationMode
fmi3ExitConfigurationMode
Reconfiguration Mode
fmi3Terminate
Terminated
a function call for this instance returns fmi3Error
fmi3FreeInstance
a function call for this or any other instance returns fmi3Fatal

Use Case:

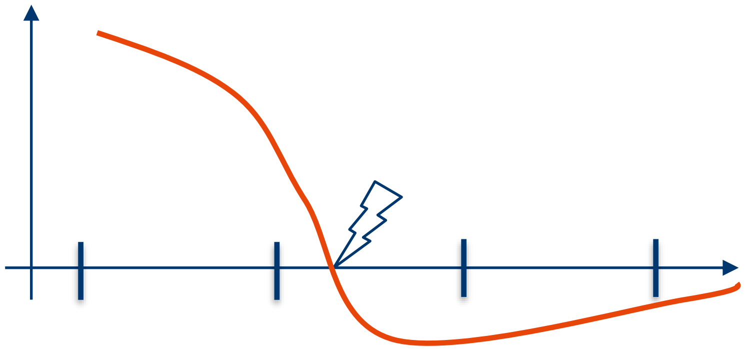
- Interrupt `fmi3DoStep()` for important internal or external events
- Exchange data with importer
- Re-initialize simulation

Result:

- Improve stability, efficiency and accuracy of Co-Simulation

Related:

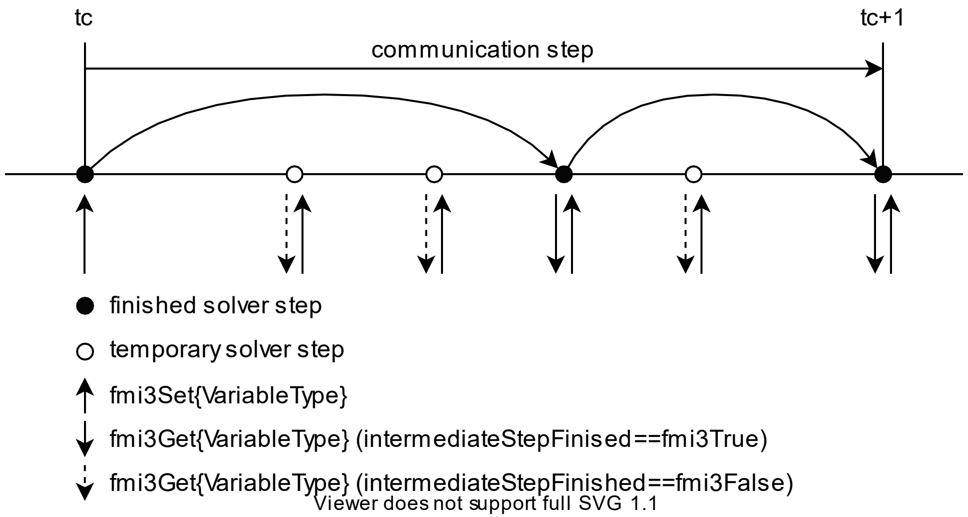
- Early-Return from `fmi3DoStep()`
- Intermediate Variable Update



FMI 3.0: Intermediate Variable Update

Use Case:

- Update inputs during Communication Step
- Allow co-simulation algorithm to interpolate inputs and outputs


tc
tc+1
communication step
● finished solver step
○ temporary solver step
fmi3Set{VariableType}
fmi3Get{VariableType} (intermediateStepFinised==fmi3True)
fmi3Get{VariableType} (intermediateStepFinished==fmi3False)
Viewer does not support full SVG 1.1

Result:

- Improve stability
- Reduce solver reinitializations

Related:

- Events for Co-Simulation
- Early-Return from `fmi3DoStep()`

FMI 3.0: Clocks

Use Case:

- Synchronize events and value exchanges across FMUs avoiding differences in time computations
- Task Scheduling within and across FMUs for Scheduled Execution

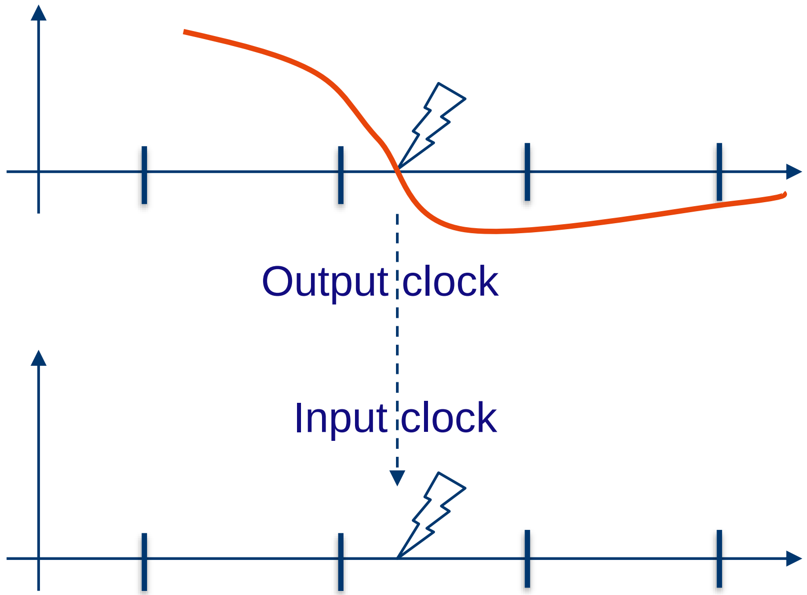
clock type		interval attribute
Time-based	**periodic clock**	constant
		fixed
		calculated
		tuneable
	aperiodic clock	changing
		countdown
Triggered	**input clock**	triggered
	output clock	triggered

Result:

- Improve stability, efficiency and accuracy of Co-Simulation

Related:

- Event Mode
- Scheduled Execution



FMI 3.0: New Types

Use Case:

- Accurate communication of internal type constraints
- Complex sensor data

Result:

- Efficient communication
- Exchange `fmi3Binary` allows non-numerical values

Related:

- Clocks

FMI 1.0/2.0	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)
	`fmi3Binary`	`const char*` for large data sets `mimeType` in modelDescription.xml
	`fmi3Clock`	Transport information about events

FMI 3.0: Array Variables

Use Case:

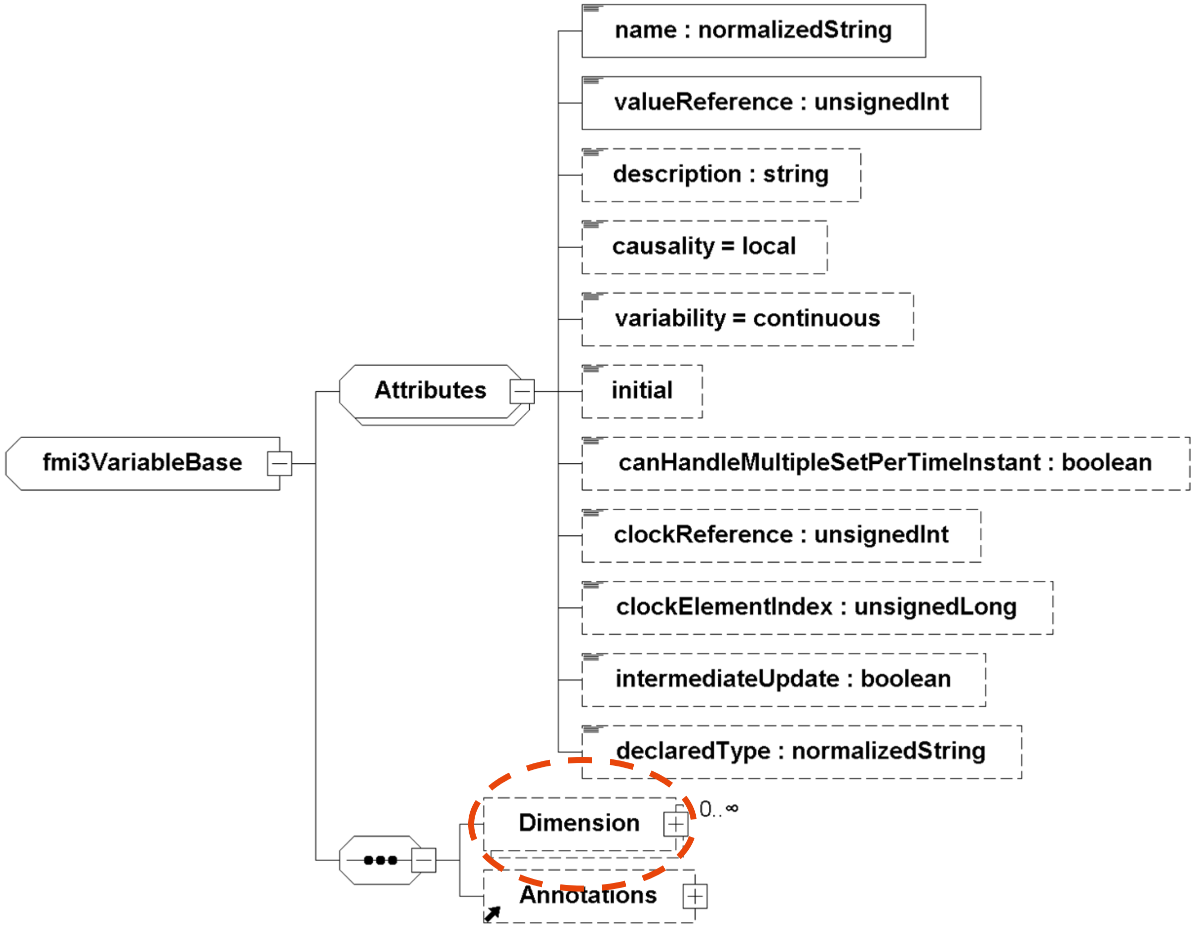
- Grouping equal interface variables

Result:

- Efficient, simplified communication
- `fmi3SetXXX`, `fmi3GetXXX` work on whole arrays

Related:

- New Types


fmi3VariableBase
Attributes
name : normalizedString
valueReference : unsignedInt
description : string
causality = local
variability = continuous
initial
canHandleMultipleSetPerTimeInstant : boolean
clockReference : unsignedInt
clockElementIndex : unsignedLong
intermediateUpdate : boolean
declaredType : normalizedString
Dimension
0..∞
Annotations

FMI 3.0: Terminals and Icons

Use Case:

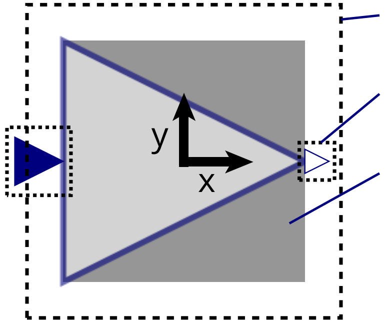
- Connect large systems correctly and graphically
- Automatically produce glue code for specific signal communications (Kirchhoff's laws, Bus structures,…)

Result:

- Semantically group signals to ease connecting compatible signals
- Graphic representations of FMUs

Related:

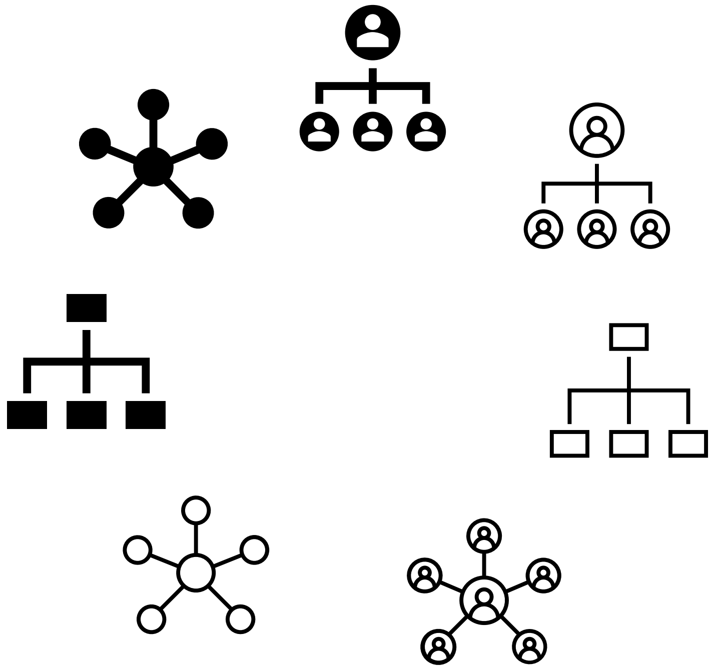
- Layered Standards



Bounding Box of FMU PNG/SVG Icon

Bounding Box of Terminal PNG/SVG Icon

Clickable Icon Size



FMI 3.0: Scheduled Execution

Use Case:

- time-discretized plant models
- task-based controller code

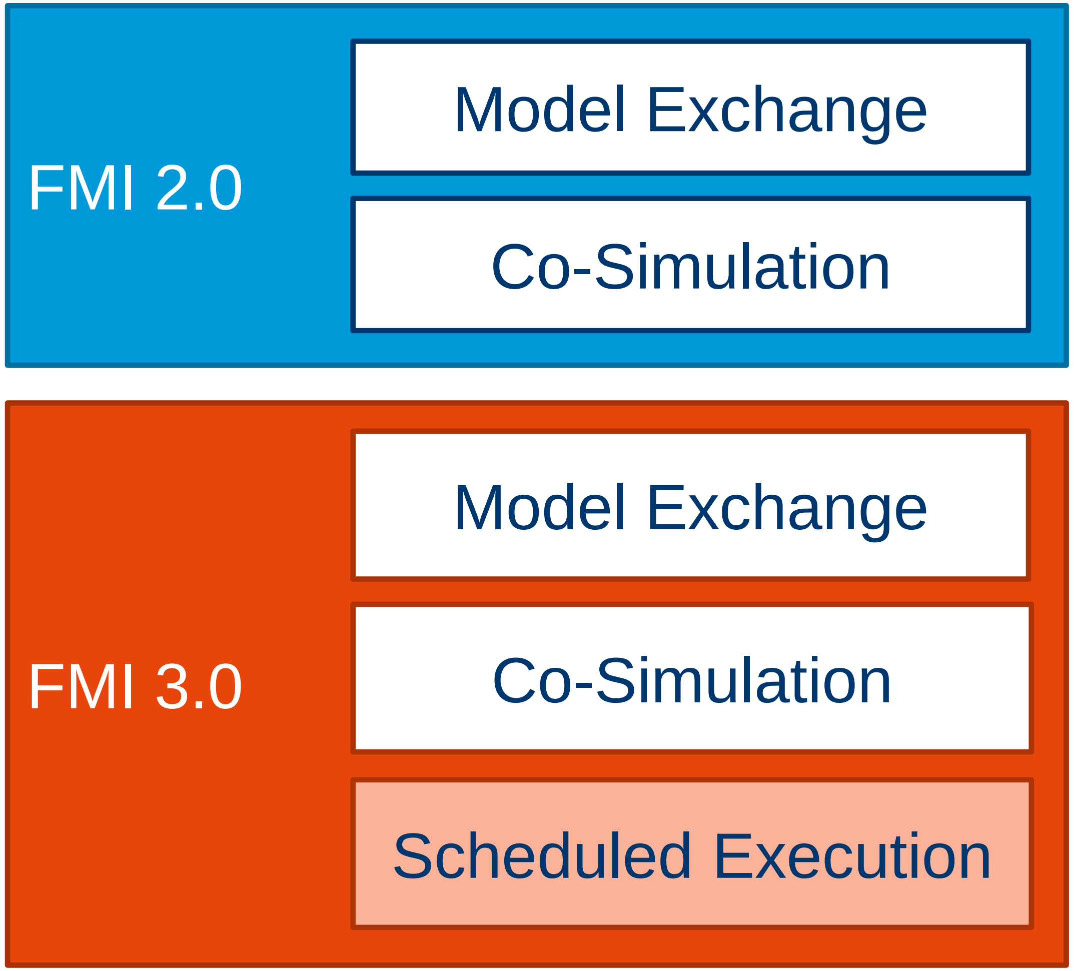
Combined from multiple development partners

Result:

- Support workshare with IP protection

Related:

- New Types



FMI 3.0: Layered Standard

Use Case:

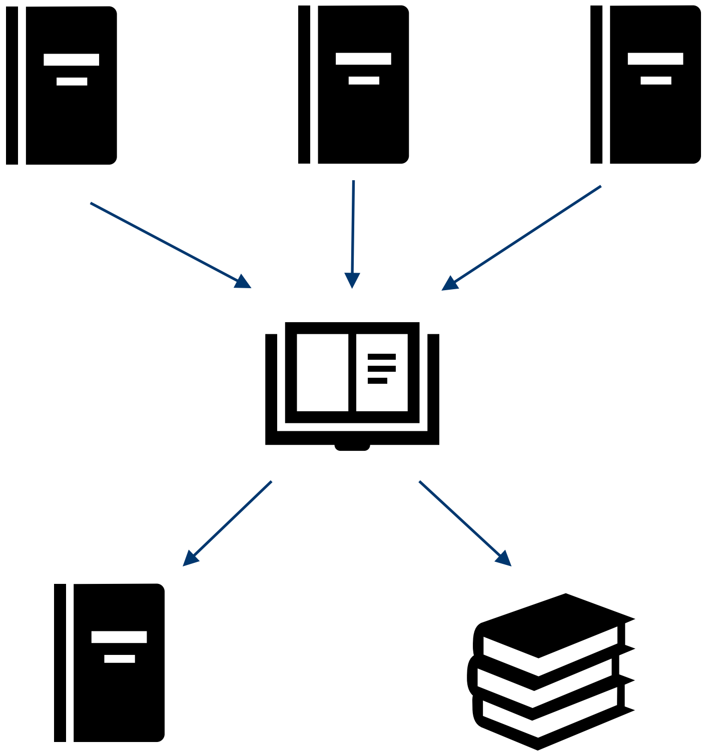
- Use FMI as base for specialized, domain-specific standards, e.g.
  - Domain specific terminal definitions
  - More description formats, like ASAM ASAP2 for vECUs

Result:

- Leverage the power of FMI, without adding complexity to FMI core standard

Related:

- Layered XCP proposal
- Layered Bus proposal



Miscellaneous

- 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.

- Improvement of specification document
  - Reuse of concepts between Model Exchange, Co-Simulation and Scheduled Execution
  - Avoid redundundancies, excessive use of links instead

Roadmap

- FMI 3.0 will be published as soon as the quality gates according to the FMI Development Process are fulfilled

- Soon the last Alpha, then Beta will be released

- Support prototype implementations by PlugFests

- Resources:
  - Development process can be tracked on GitHub: https://github.com/modelica/fmi-standard
  - FMPy is permanently updated to support FMI 3.0: https://github.com/CATIA-Systems/FMPy
  - Reference FMUs: https://github.com/modelica/Reference-FMUs