Merging FMI & MBSE

Designs, Experiences & Future Possibilities

Nils Paulsson, Ph.D.

Saab Aeronautics

Merging FMI & MBSE

FMI/FMU 2.0 + xtUML = Co-simulation demonstrator

FMI – Functional Mock-up Interface (Interface-standard for a simulated entity)

FMU – Functional Mock-up Unit (A simulated entity)

xtUML – Executable UML (Formalized MBSE methodology)

MBSE – Model Based Systems Engineering

Merging FMI & MBSE

Co-simulation demonstrator

- Uses *System Monitoring* (e.g. cockpit temp) as demo simulation
- Modeled in xtUML
- Provides FMI 2.0 interface for integration in a simulation infrastructure

Done as part of the **OpenCPS** project

- European ITEA3 research project
- Interoperability between Modelica – UML – FMI/FMU

The Design - Domain model

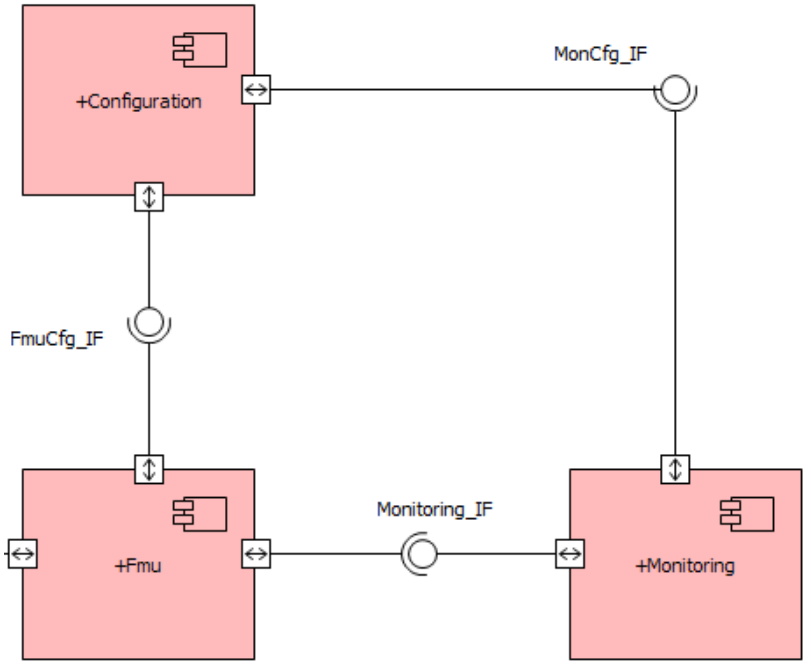
Domain

A distinct *subject matter* with its own vocabulary, rules, constraints and behavior.

A concern in its own rights.

Domains

Provides configuration information.



Executes simulation. Manages sim-variables. Talks with sim-master.

Monitors system signals.

The Design - Bridges

Bridges

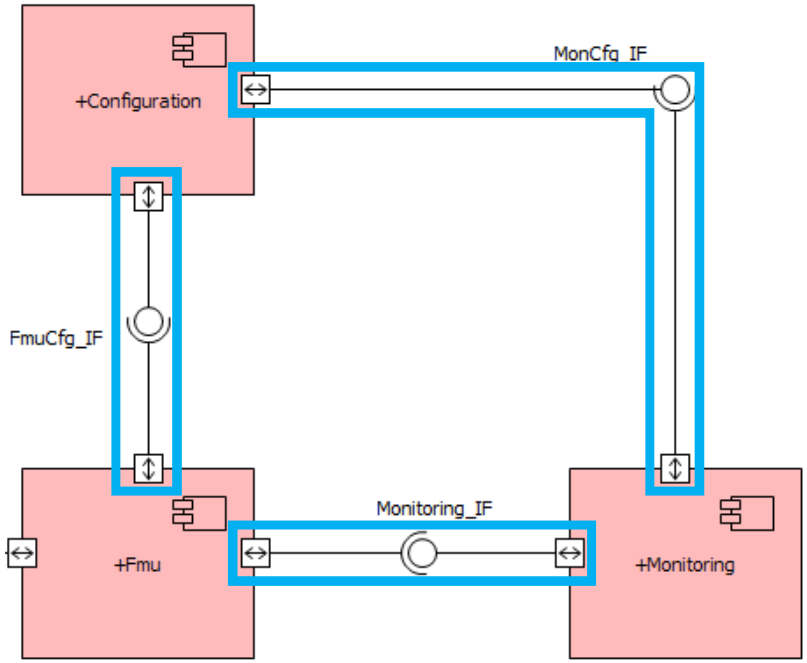
A duplex translator between 2 domains.

(Translates vocabulary, rules, concepts, etc.)

A duplex device driver adhering to the *Dependency inversion principle**.

* Check Wikipedia

Bridges



The FMU – Monitoring Bridge

Bridge translations from FMU

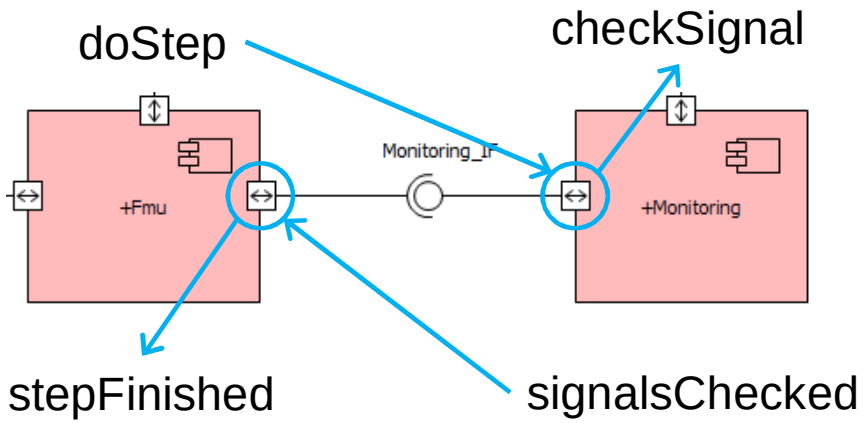
Events

- doStep -> checkSignals

Bridge translations to FMU

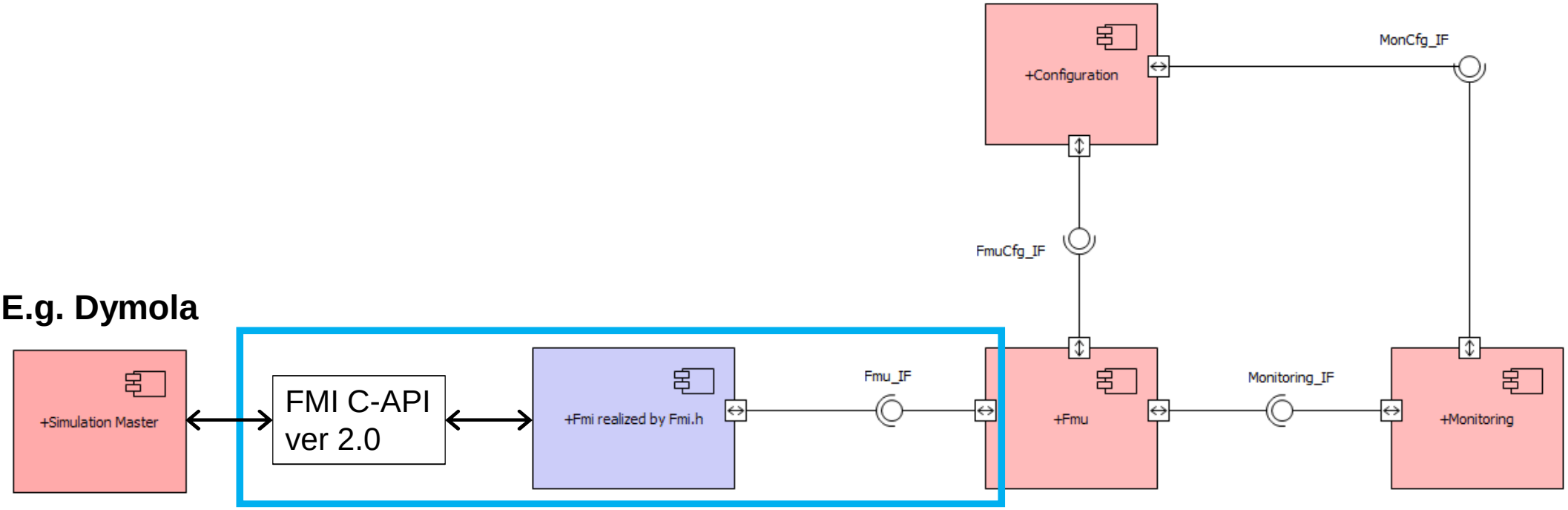
Events

- signalsChecked -> stepFinished



The FMU-FMI Bridge

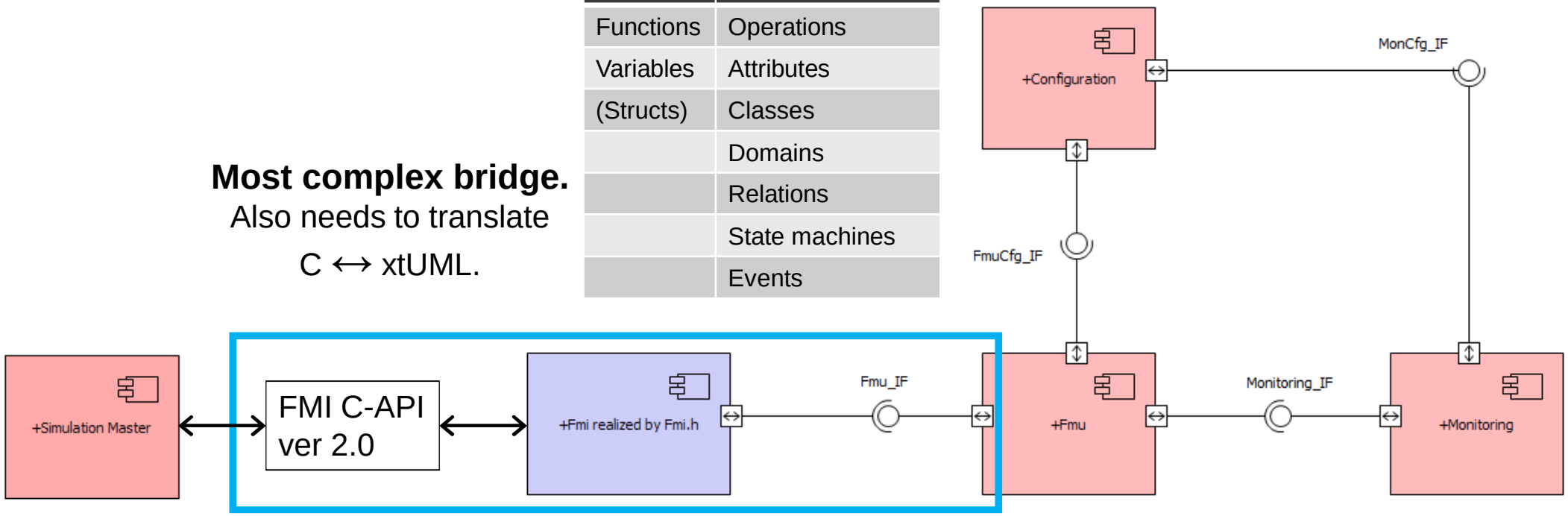
E.g. Dymola



The FMU-FMI Bridge

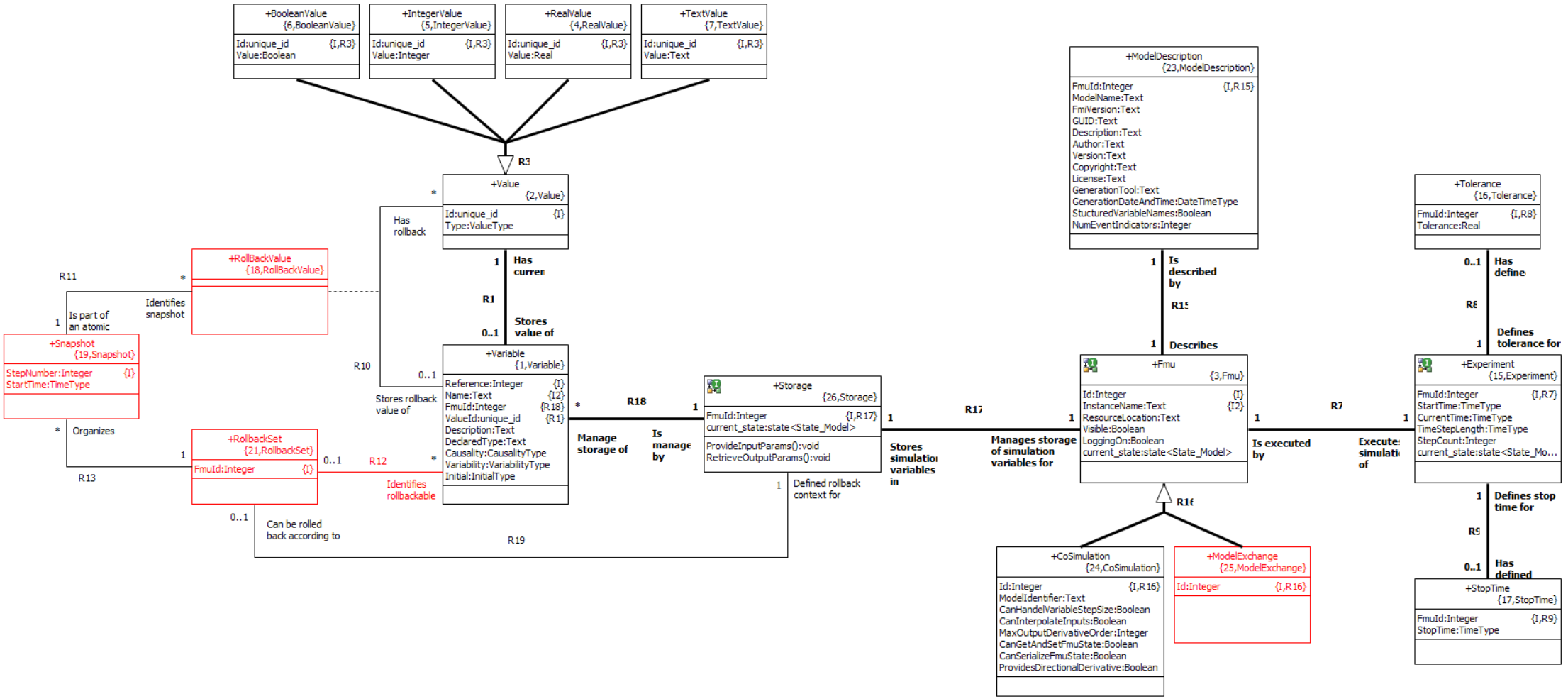
Most complex bridge.
Also needs to translate
C ↔ xtUML.

C	xtUML
Functions	Operations
Variables	Attributes
(Structs)	Classes
	Domains
	Relations
	State machines
	Events

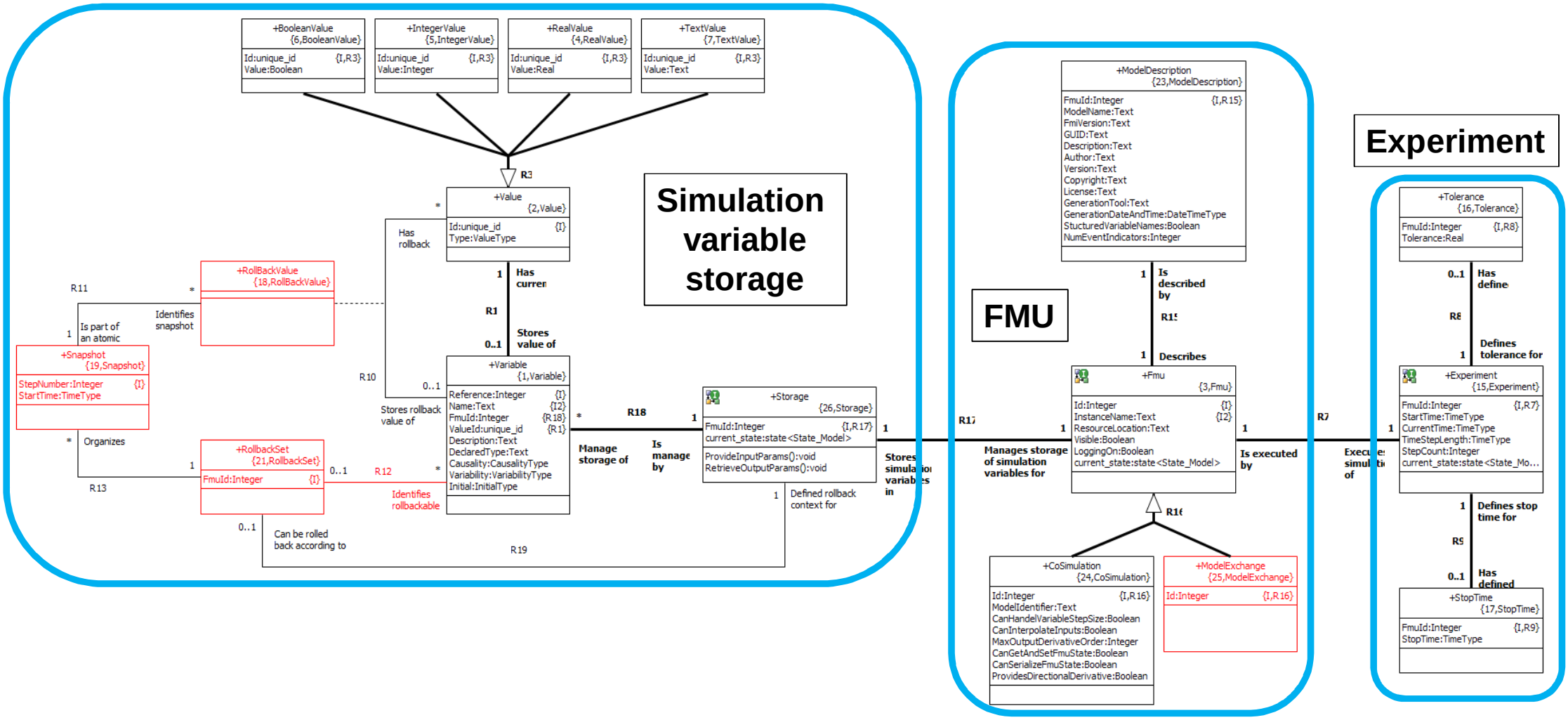


The Design - The FMU domain

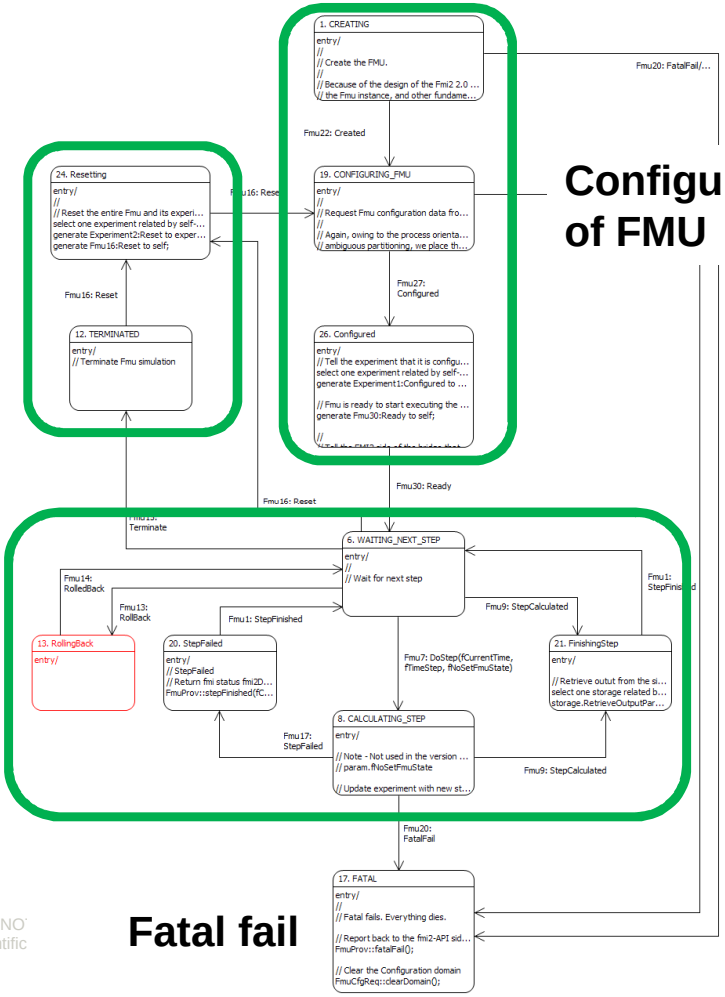
The FMU domain



The FMU domain



The FMU class' life cycle *(state machine)*



Reset & Terminate

Configuration of FMU

Simulation execution

Fatal fail

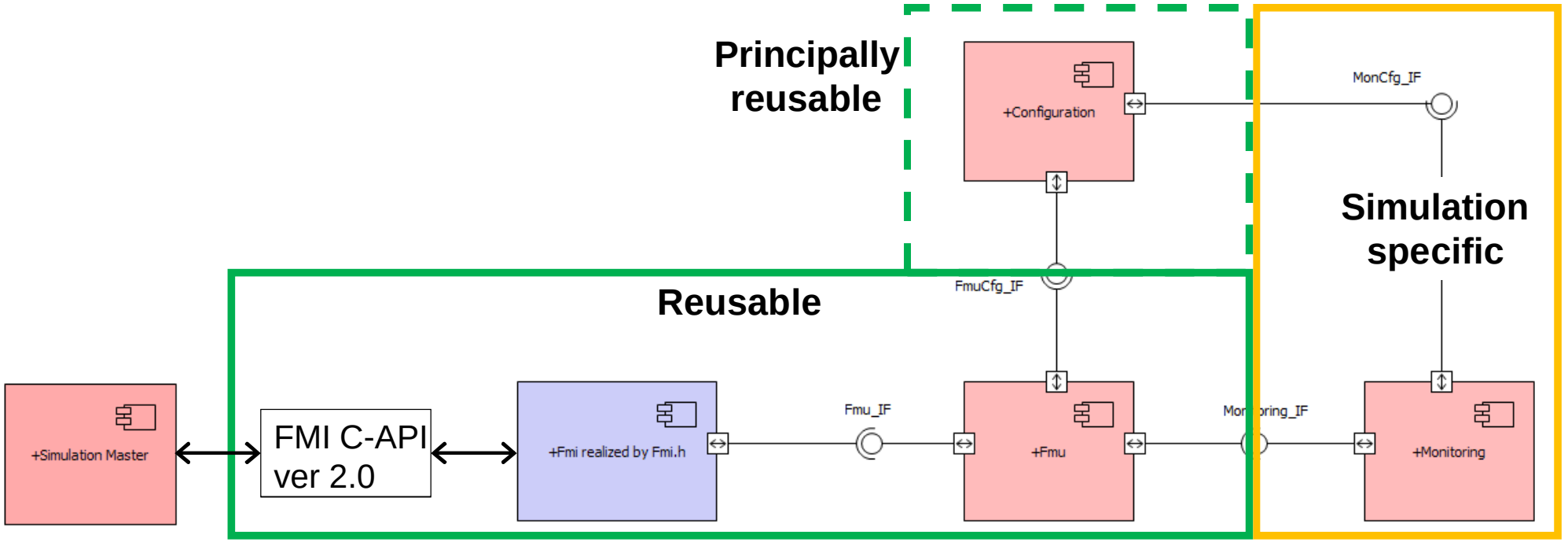
Experiences & future possibilities

Experiences

- The FMU-model successfully generated & compiled into a DLL exposing an FMI 2.0 interface.
- Successfully simulated the *System Monitoring*-FMU using Dymola and OMSimulator.
- I.e. It works!

Experiences

- Significant parts of the xtUML-FMU is reusable.

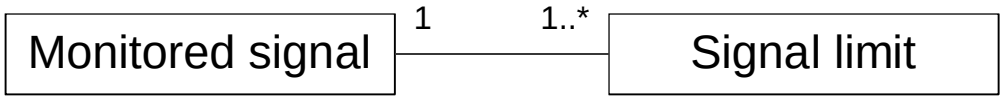


Experiences

- FMI is *Process* oriented – xtUML *System* oriented (i.e. MBSE)
- In/out parameters are isolated signals

General systems theory*:

*System: An organized entity made up of **interrelated** and **interdependent** parts.*



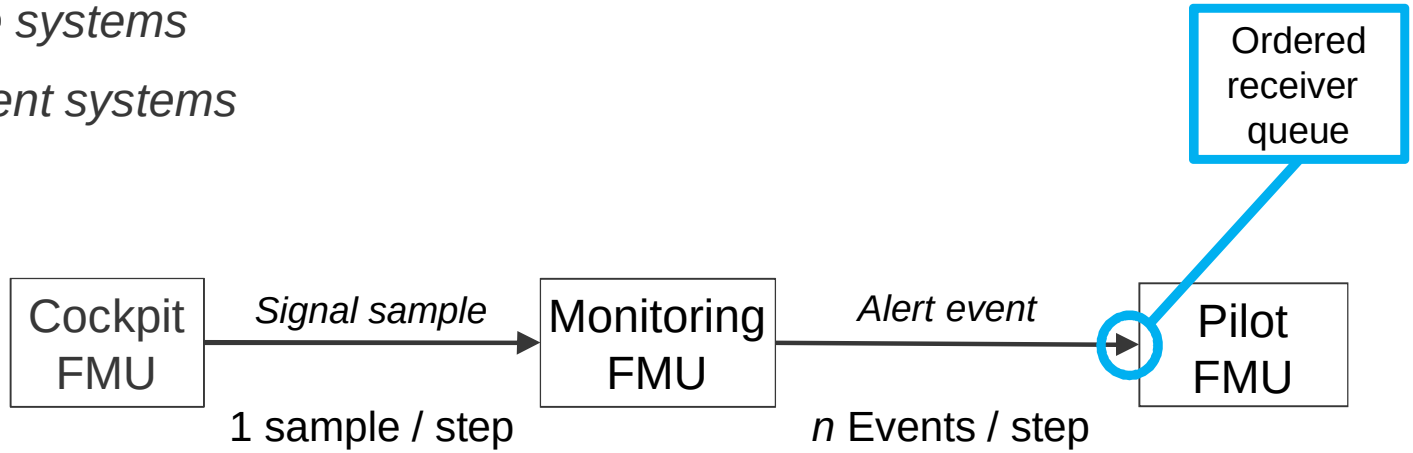
*E.g. Ludwig von Bertalanffy

Experiences

FMI only supports *Parameter* driven simulations - not *Event* driven.

Why events?

- *Reactive systems*
- *Concurrent systems*



Antony Hoare, Communicating Sequential Processes
Leslie Lamport, Coherence in Distributed Systems

Experiences

Two different beasts

Parameter

- Has a type.
- Has a value.
- Not always well-defined (in a simulation context).
- Simple life cycle (Create – Assign – Delete).

Event

- Has a signature.
- Has a source.
- Has a destination.
- May have payload.
- Time stamped / ordered.
- Is flank triggered.
- Is well defined once sent.
- Complex life cycle (Create – Send – Transmit – Receive – Consume – (React upon) – Die).

Experiences

The FMI standard has some issues with *Separation of concerns**.

- Memory management of *Text* parameters differs from that of *Integers*, *Reals* and *Booleans*
- Configuration information provided by both the FMI API and *modelDescription.xml*.
- The exact point to calculate initial output is ambiguous.
  (Suggest adding *fmi2CalcInitialValues()* to the API)
- Some state charts in the 2.0 standard documentation need attention

* Check Wikipedia

Conclusion

The FMI standard supports a generic, reusable, pluggable xtUML-model of an FMU.

The FMU-model has been successfully simulated using Dymola & OMSimuator.

The current FMI standard (2.0) would improve by a bit of MBSE treatment (Simulation of *Systems*).

Thank You.