Cyber-Physical Systems Engineering: An INTO-CPS Tutorial

Peter Gorm Larsen, Kenneth Lausdahl, Aarhus University

John Fitzgerald, Ken Pierce, Newcastle University

Horizon 2020 Programme

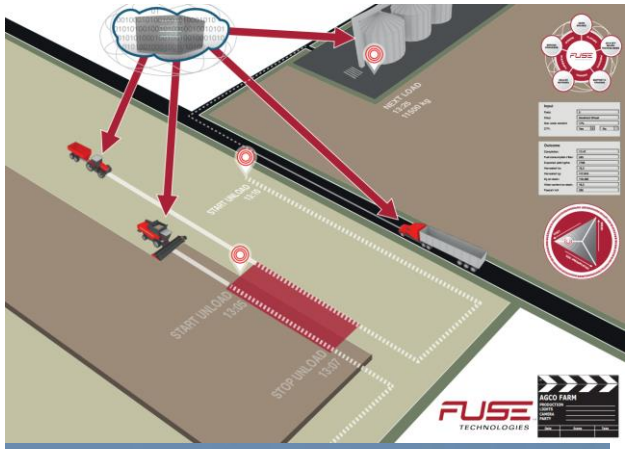
Tutorial Schedule

- 09:00 – 10:00: Introduction to INTO-CPS (PGL)
- 10:00 – 10:30: Coffee break
- 10:30 – 11:30: Introduction to VDM (JF)
- 11:30 – 12:30: Introduction to 20-sim (KP)
- 12:30 – 13:30: Lunch
- 13:30 – 14:30: Industrial results using INTO-CPS (PGL)
- 14:30 – 15:30: Demonstration of INTO-CPS Capabilities (KL+KP)
- 15:30 – 16:00: Coffee break
- 16:00 – 17:00: Demonstration of Hands-on Practicals

What is a Cyber-Physical System?

INTO-CPS

- Systems of interacting systems
  - Computing elements
  - Physical elements
  - Human interactions
- Complex, networked character
- Distributed control
- Error detection and recovery



Our position in a nutshell

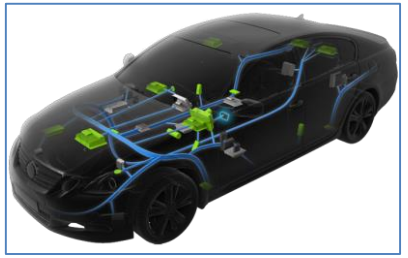
INTO-CPS

We advocate:

- Cyber-Physical Systems Engineering
  - The product is a system: software is not the end!
- Multidisciplinary collaborative modelling
- (Co-) simulation as well as verification
  - Promotes Design Space Exploration
  - Entails well-founded co-simulation orchestration

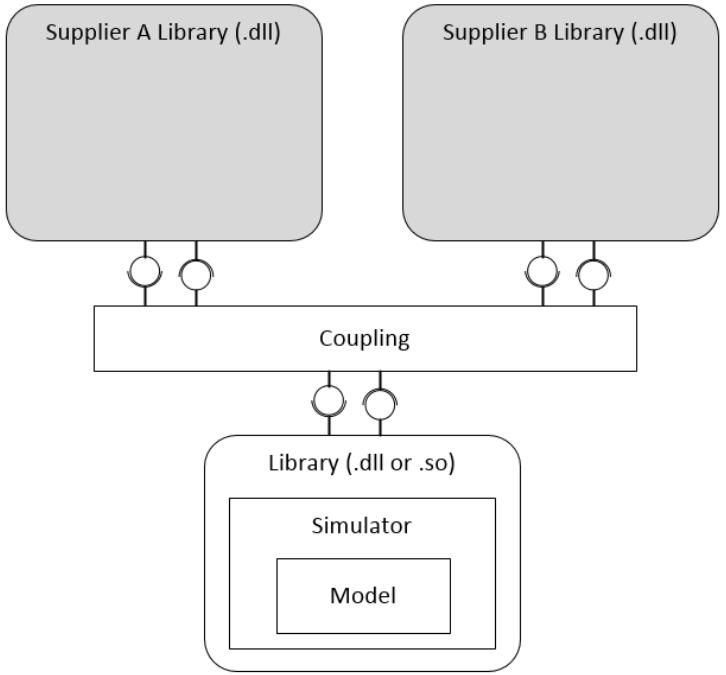
Our approach:

- Co-simulation of multiple Discrete Event and Continuous Time models
- A tool chain – not single tools
  - Requirements and Architectural models (in SysML)
  - Traceability support through development
- FMI interfaces constituent models
- Semantic foundations in Unifying Theories of Programming



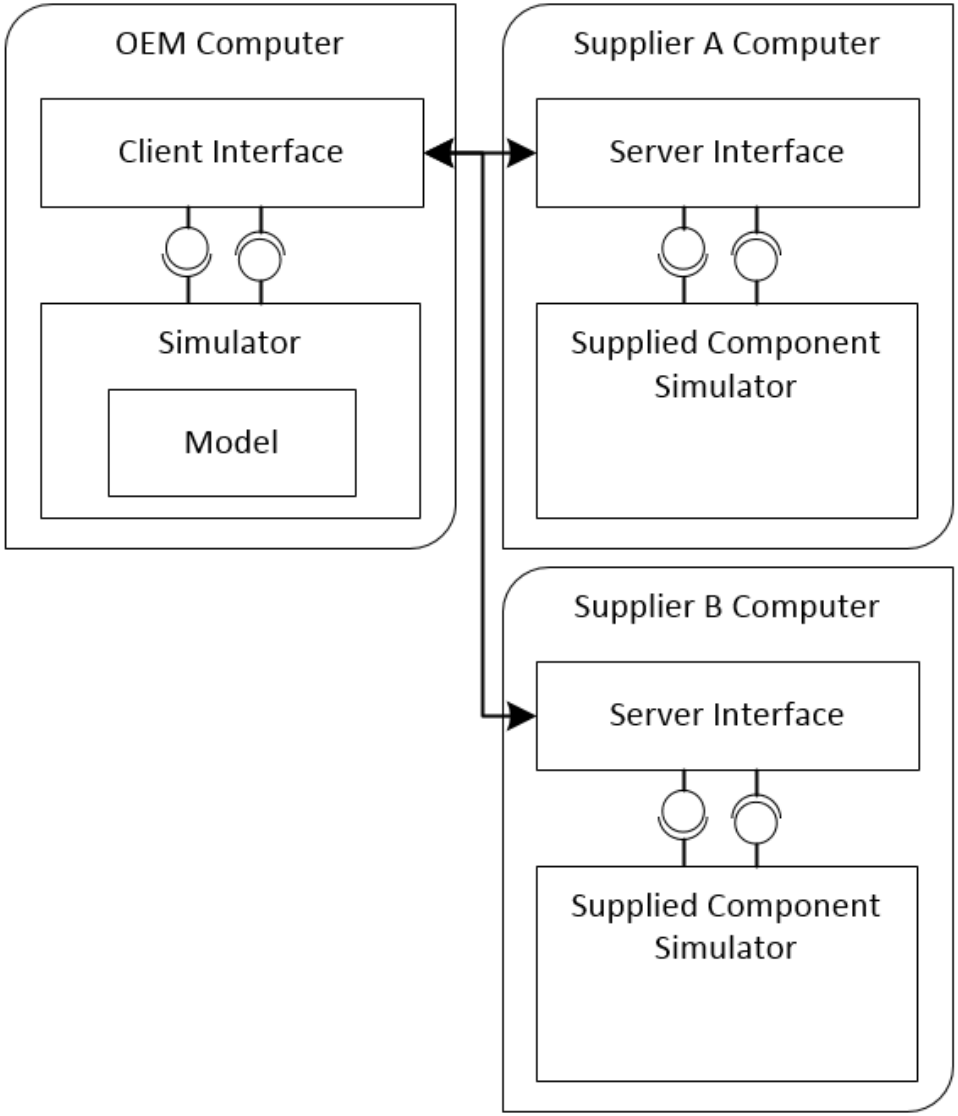
What is Co-simulation?

- Simulation of a system
- Coupling of multiple simulators
- Optionally as black-boxes
- Each simulating one or more models
- Built with different formalisms/tools.
- Co-simulation scenario
- Description of the system
- The simulators and their dependencies
- Data about the capabilities of each simulator.



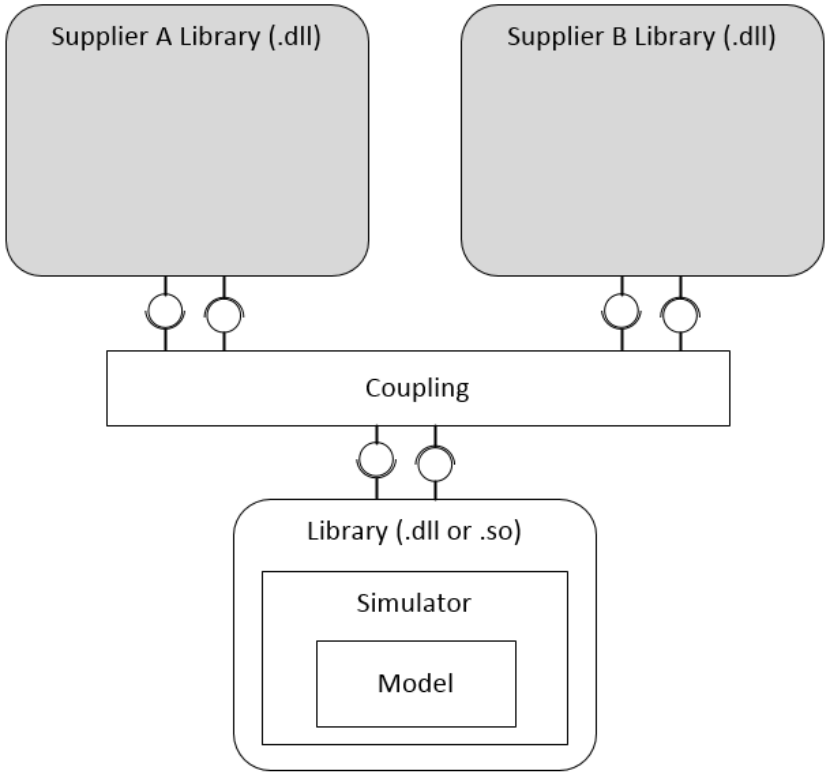
Remote Black-box Simulators

- Suppliers make a simulator available through an API
- Integrator takes care of programming an interface
- Good IP Protection
- Different suppliers require different interfaces

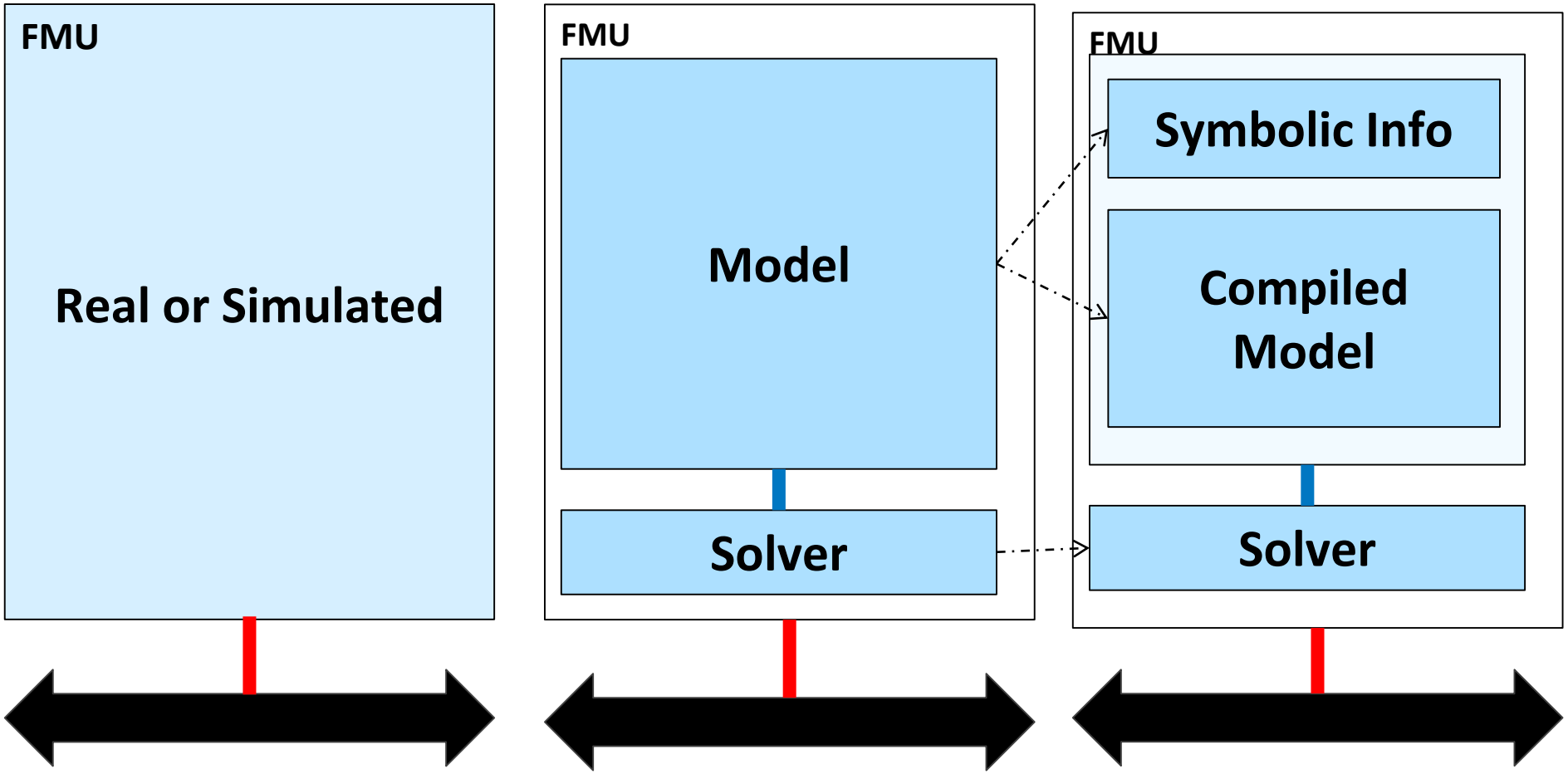


Functional Mock-up Interface Standard

- Simulator and model exported as a standardized C library
- Standard interaction with any simulator
- Every simulator is a black box.
- Executed locally but can communicate with a remote server

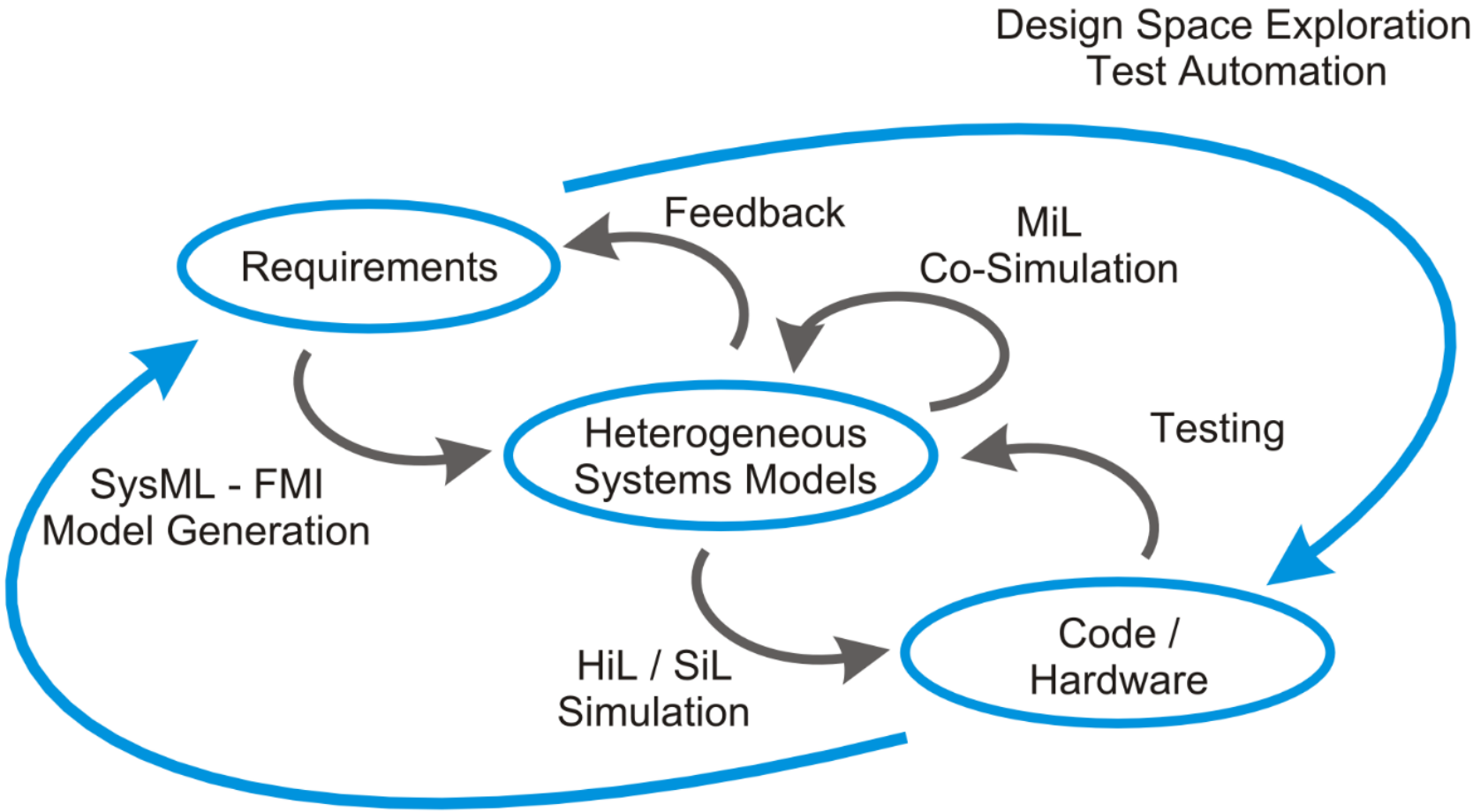


Inside an FMU



A New Toolchain for CPS Design

INTO-CPS



INTO-CPS Objectives

1. Build an open, well-founded tool chain for multidisciplinary model-based design of CPS that covers the full development life cycle of CPS
2. Provide a sound semantic basis for the tool chain
3. Provide practical methods in the form of guidelines and patterns that support the tool chain
4. Demonstrate in an industrial setting the effectiveness of the methods and tools in a variety of application domains
5. Form an INTO-CPS Association to ensure that project results extend beyond the life of the project

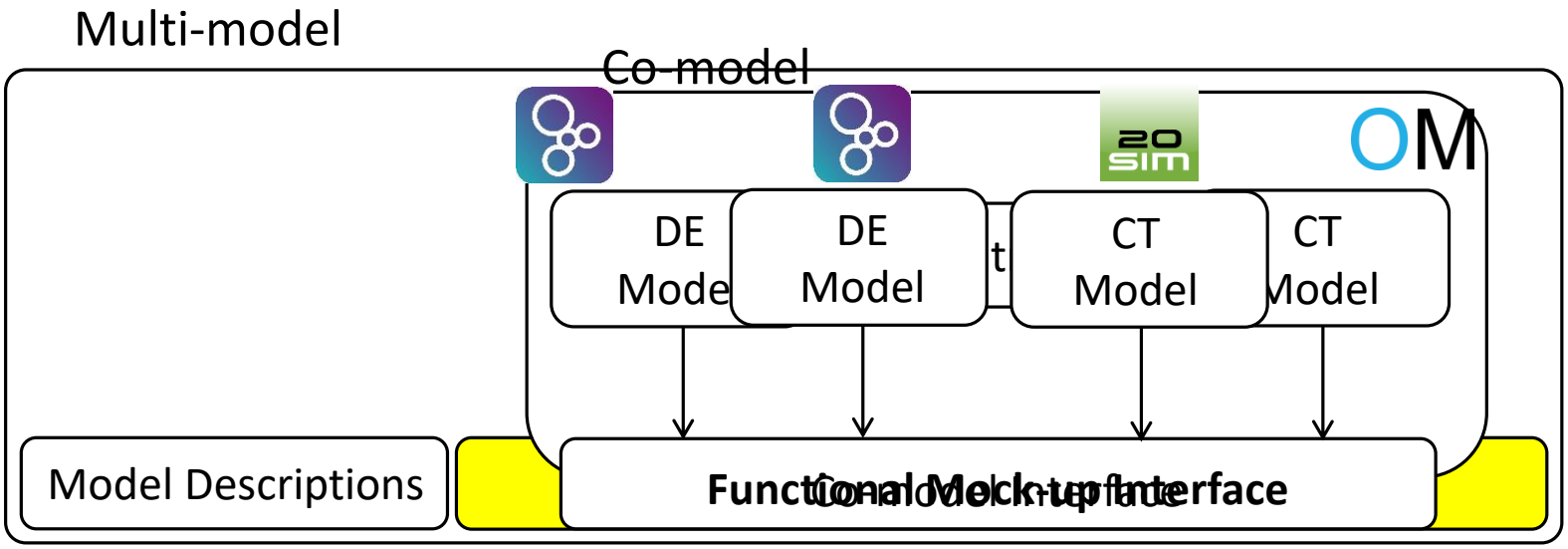
INTO-CPS Unique Selling Points

1. Faster route to market for engineering CPSs
2. Avoiding vendor lock-in by open tool chain
3. Exploring large design spaces efficiently
4. Limiting expensive physical tests
5. Enabling traceability for all project artefacts

CPS Engineering Needs

- Enable collaboration across disciplines
- Keep development costs low
- Keep time-to-market short
- Explore the complex design space efficiently
- Ensure tolerance against “nasty” faults
- Build up documentation for the working solution
- Provide confidence to external stakeholders

Co-modelling to Multi-modelling



Modelio — SysML modelling

Overture — Discrete-event modelling

20-sim — Continuous-time and physical-systems modelling

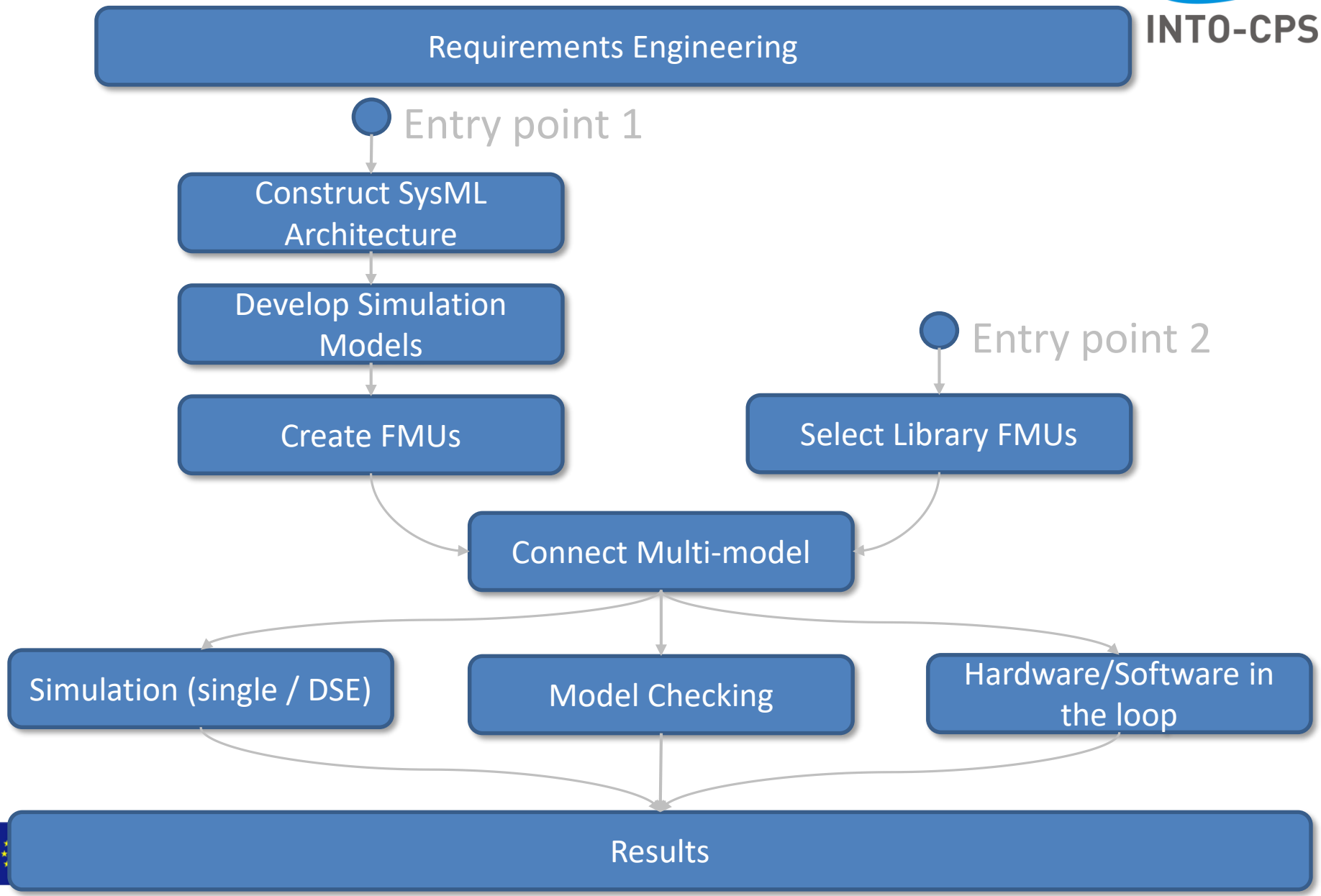
OpenModelica

Crescendo — Co-simulation solutions

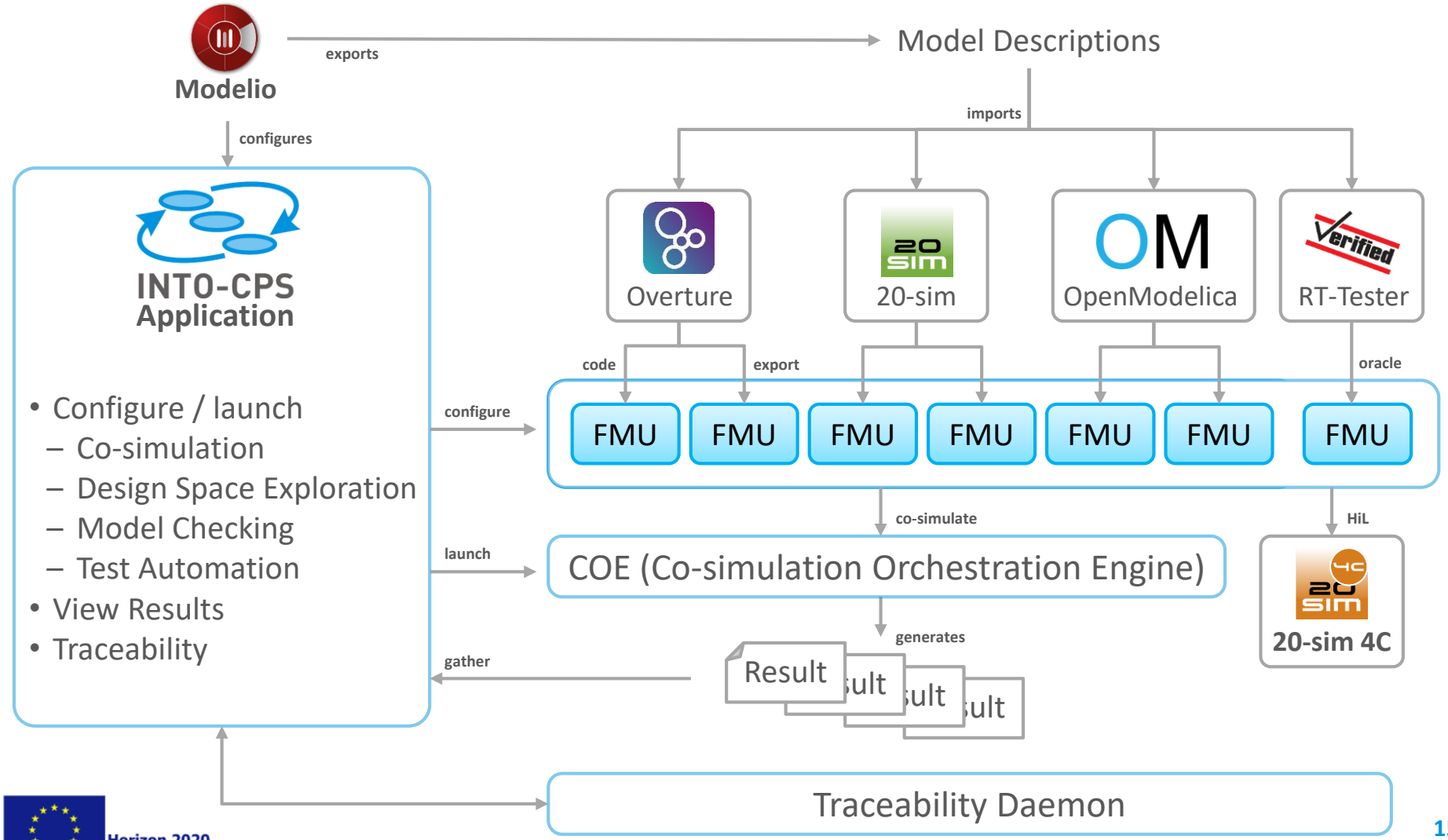
TWT Engine

RT-Tester — Test automation / model checking

Outline Work Flow



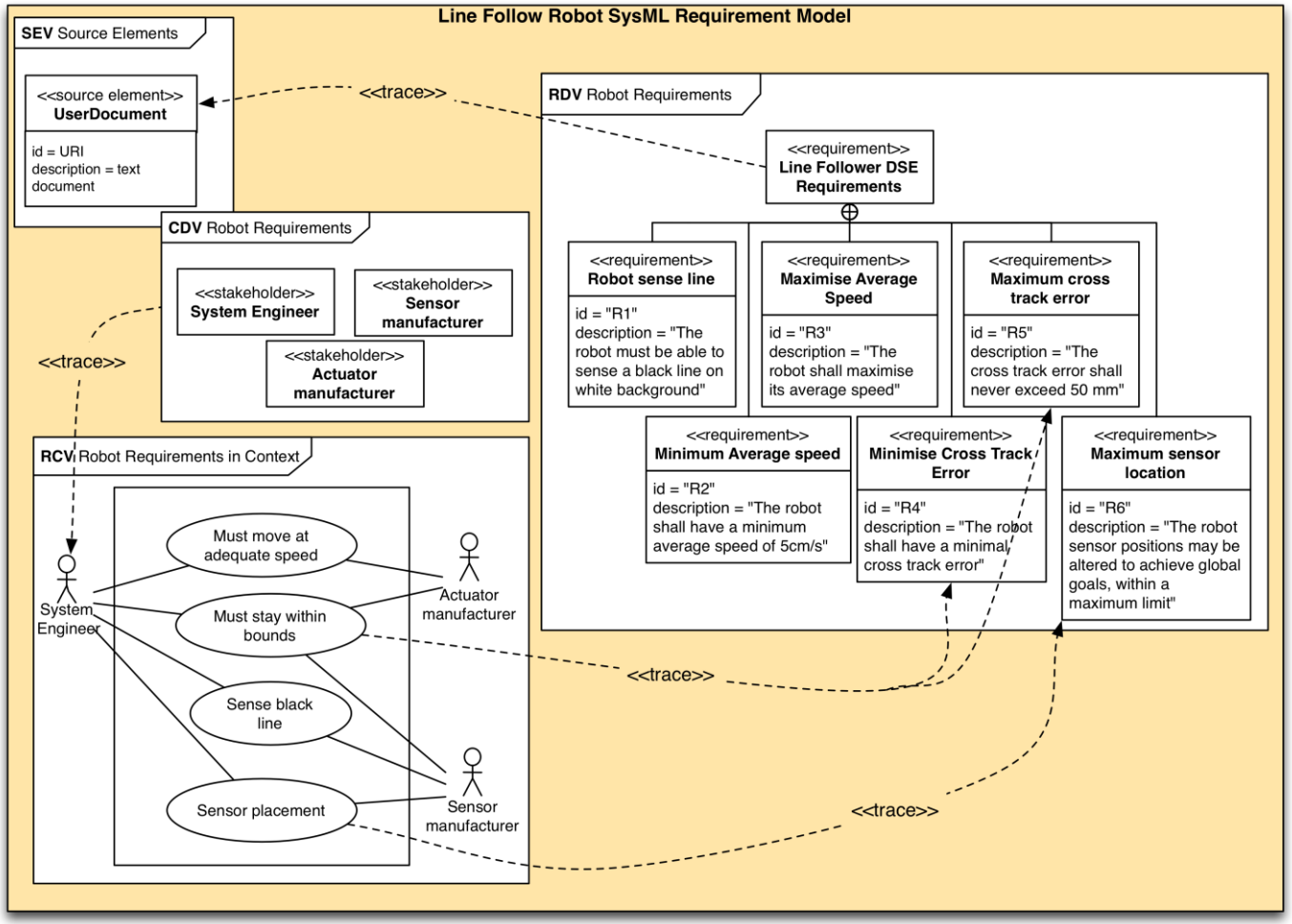
The INTO-CPS Tool Chain



Co-simulation engine

- Fully FMI 2.0 compliant Master Algorithm
- Support for discrete event (DE) and continuous time (CT) models, using proposed FMI extensions
- Multi-platform, 32/64 bit (Java-based)
- Application based on Electron (web-technology)
- Fixed and variable step size algorithms
- FMI 2.0 Import/Export created for Overture, OpenModelica, 20-sim
- Has also been tested with:
  - Dymola
  - Modelon FMI Toolbox for MATLAB/Simulink
  - 4DIAC
  - SimulationX
  - Unity

INTO-CPS Methodology



Requirement Engineering

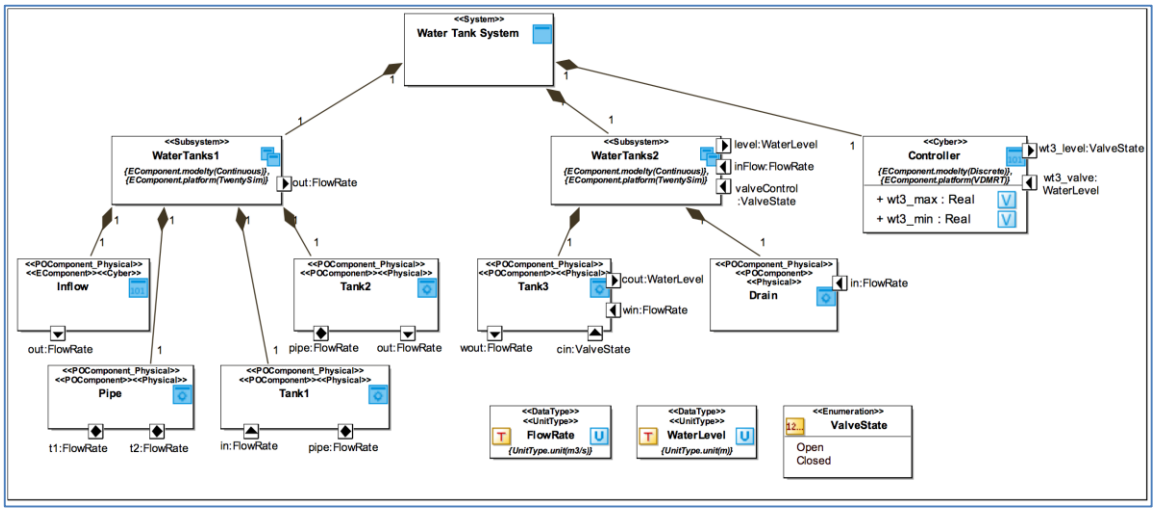
Architectural Modelling

Simulation Modelling

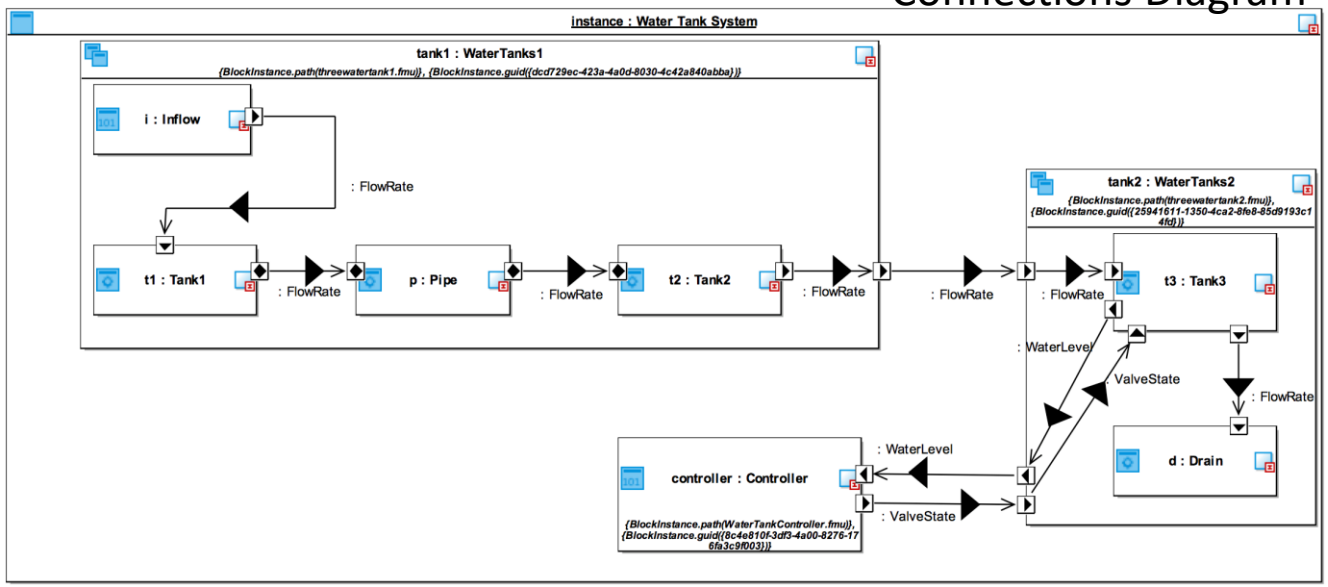
Analysis

Architectural Modelling

Architecture Structure Diagram



Connections Diagram



Requirement Engineering

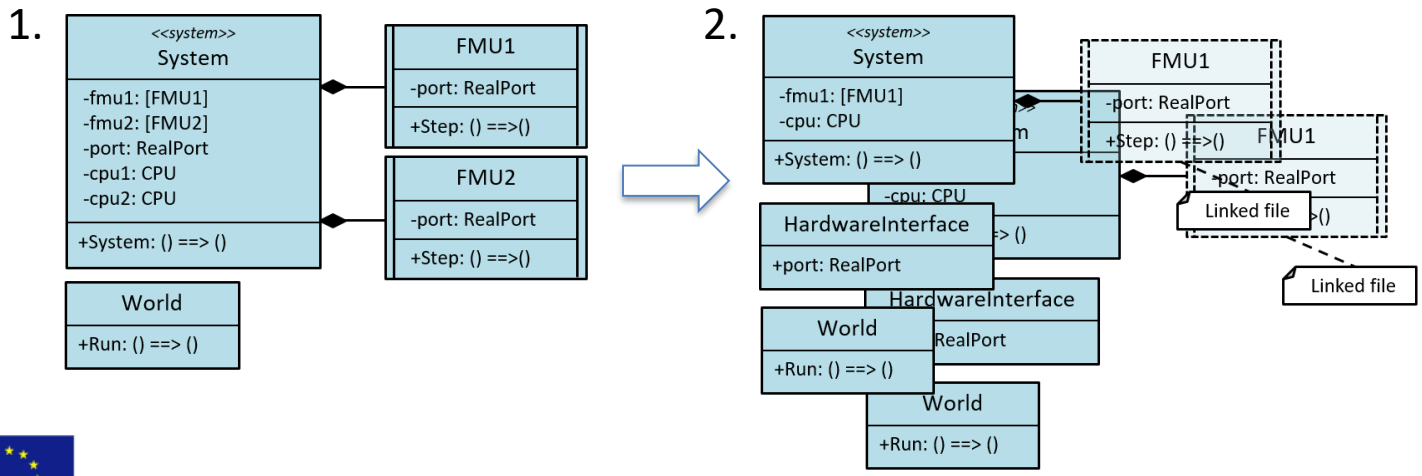
Architectural Modelling

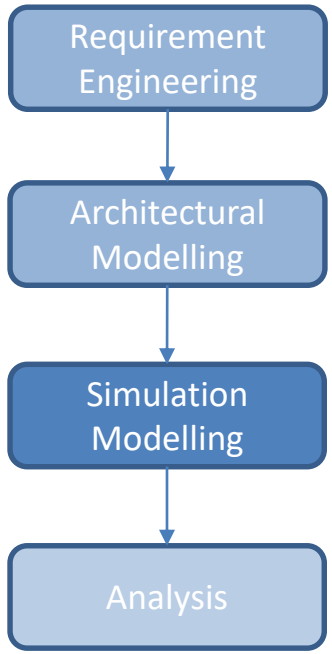
Simulation Modelling

Analysis

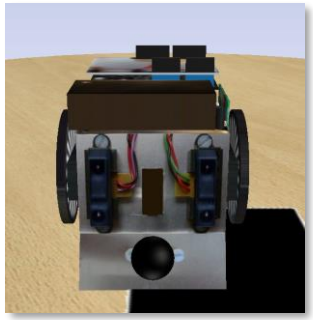
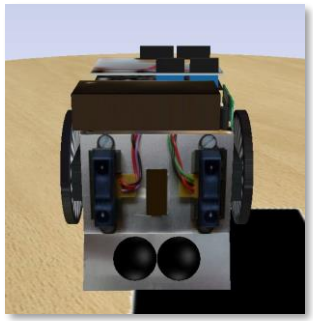
Simulation Modelling

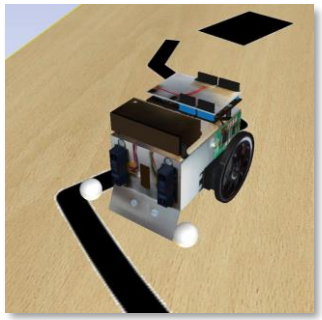
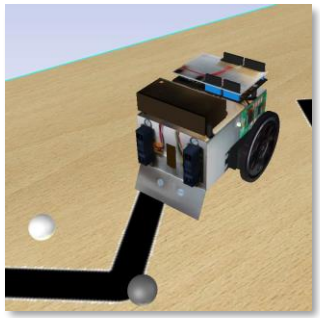
- "DE-first" approach, used in IPP4CPPS
  1. Model and test entire system in DE
     - One class per EComponent, both DE and (abstract) CT
     - Use Ether pattern to connect classes if necessary
  2. Move to multi-model
     - One project per EComponent (link class definition)
     - Export FMUs





Design Space Exploration

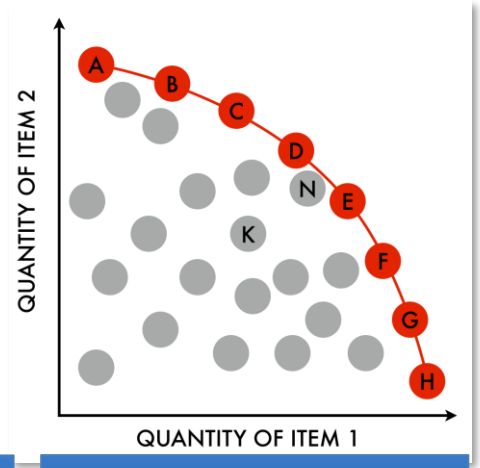



How to Explore?

- Exhaustive
- Taguchi Methods
- Space Culling
- Genetic Methods

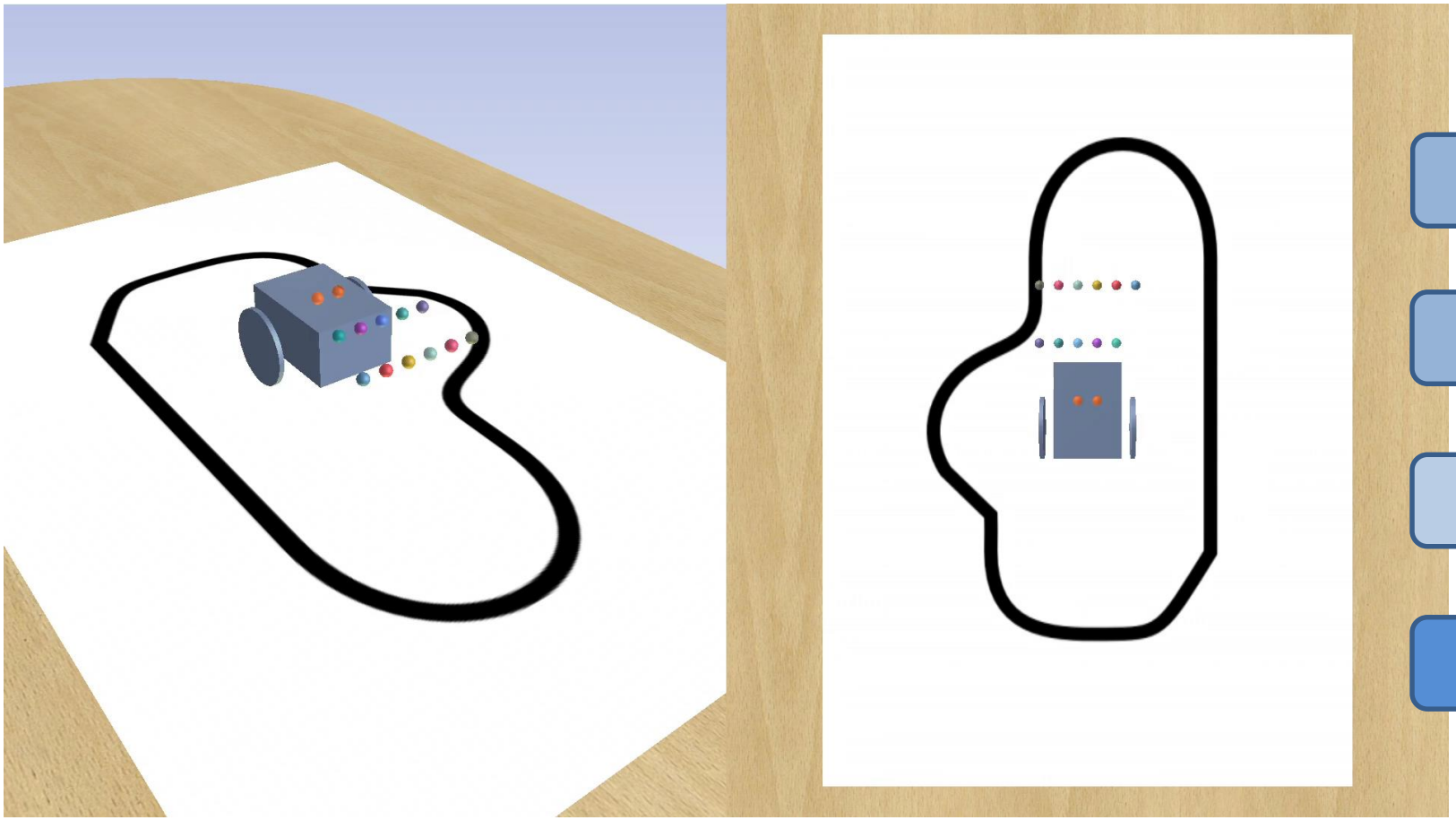
How to Assess?

$$V_a = w_1^a v_1^a(x_1^a) + w_2^a v_2^a(x_2^a) + ... + w_n^a v_n^a(x_n^a)$$



DSE Driver	Objective Evaluator	Ranking	Presentation
Simulation parameters, control of process	Objective measures and constraint satisfaction from raw simulation results	Ranking designs according to objective values	Display of results and search progress to the user

Example DSE Analysis



Requirement Engineering

Architectural Modelling

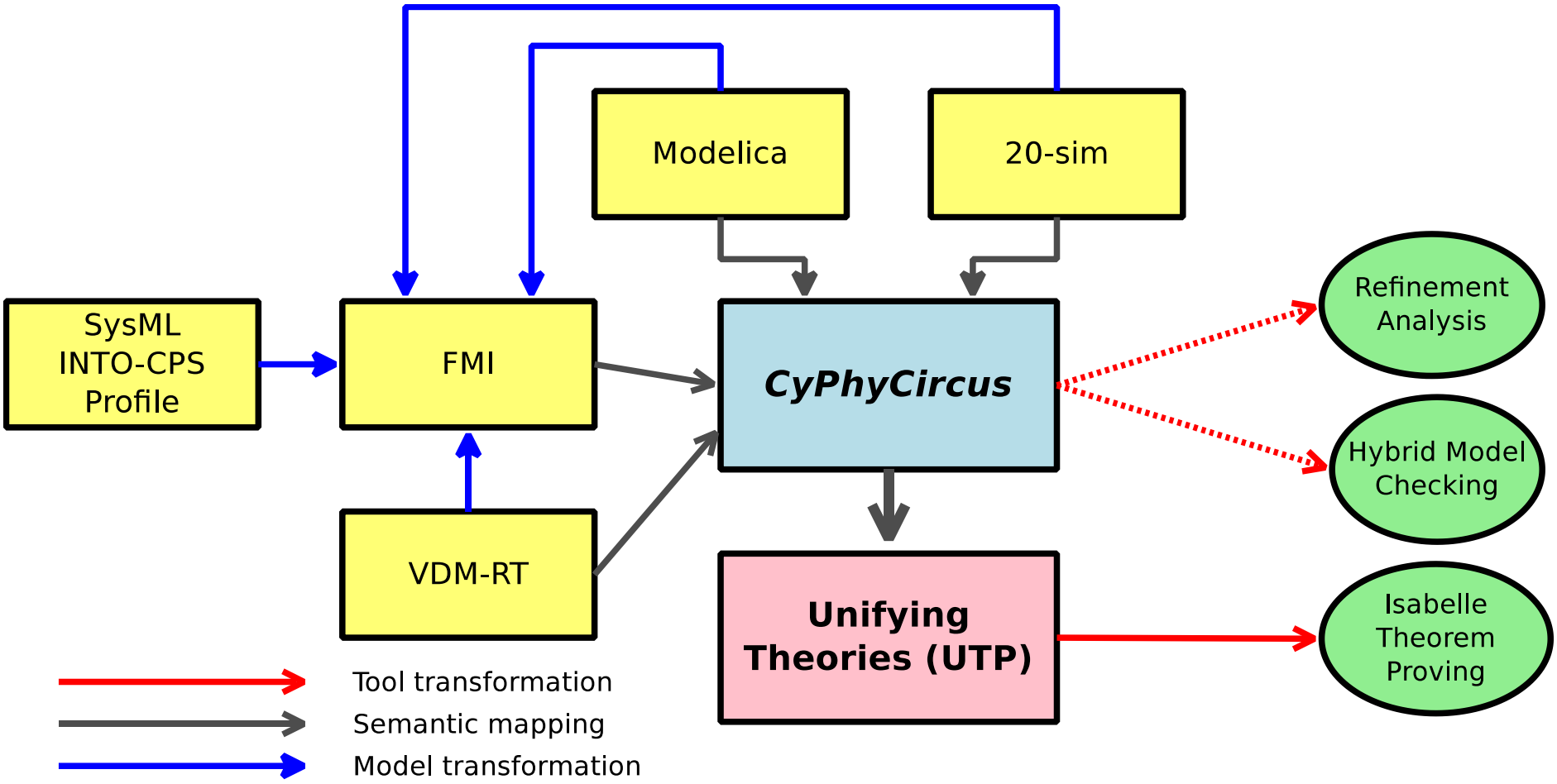
Simulation Modelling

Analysis

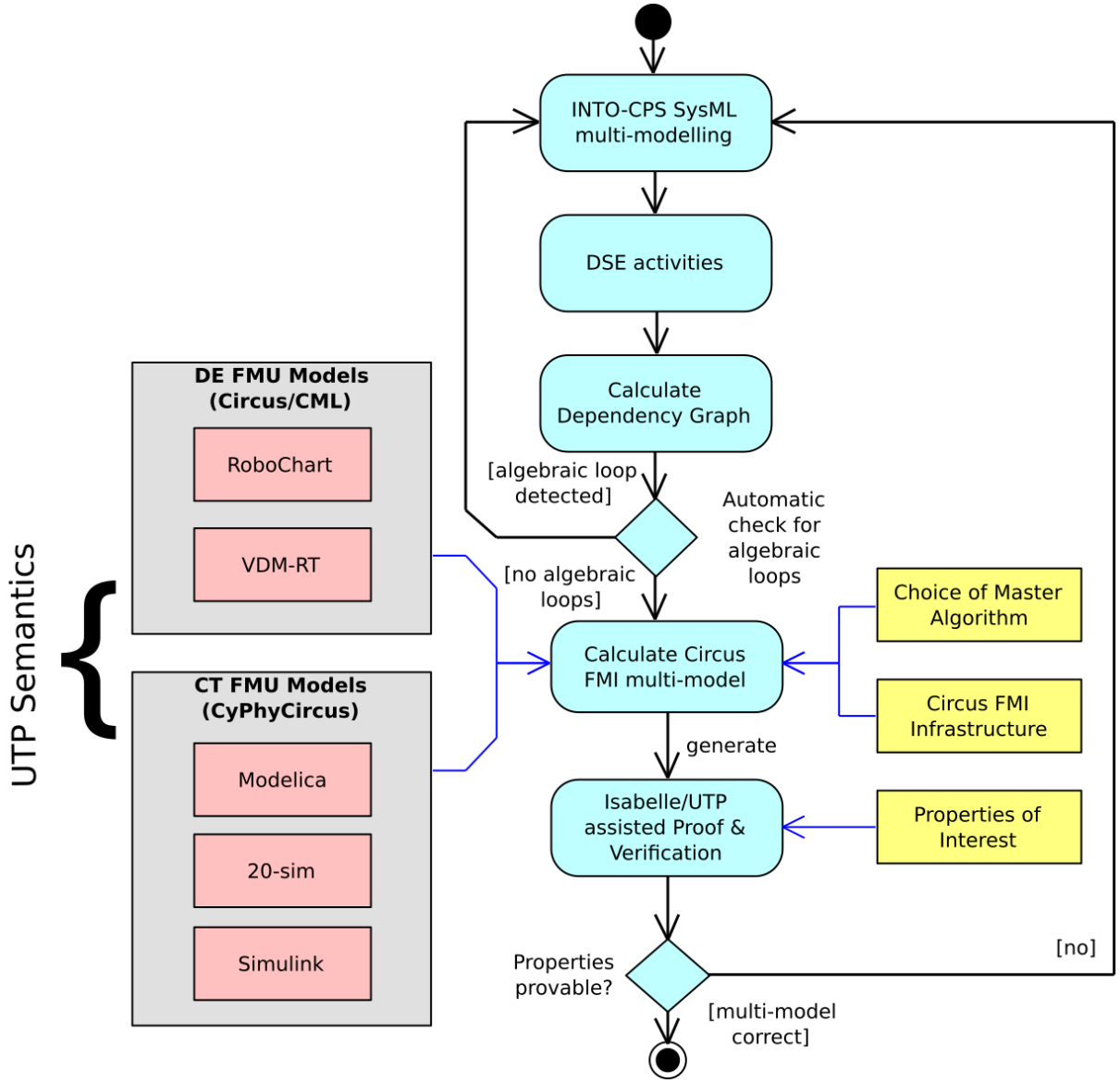
Co-Simulation Foundations

- Initial foundations developed for
  - SysML
  - VDM-RT
  - Modelica
  - FMI
- SysML CPS profile defined
  - Architecture Structure Diagram
  - Connections Diagram
  - Design Space Exploration Diagram
  - Support for units in the exported model descriptions

Common Semantics using UTP

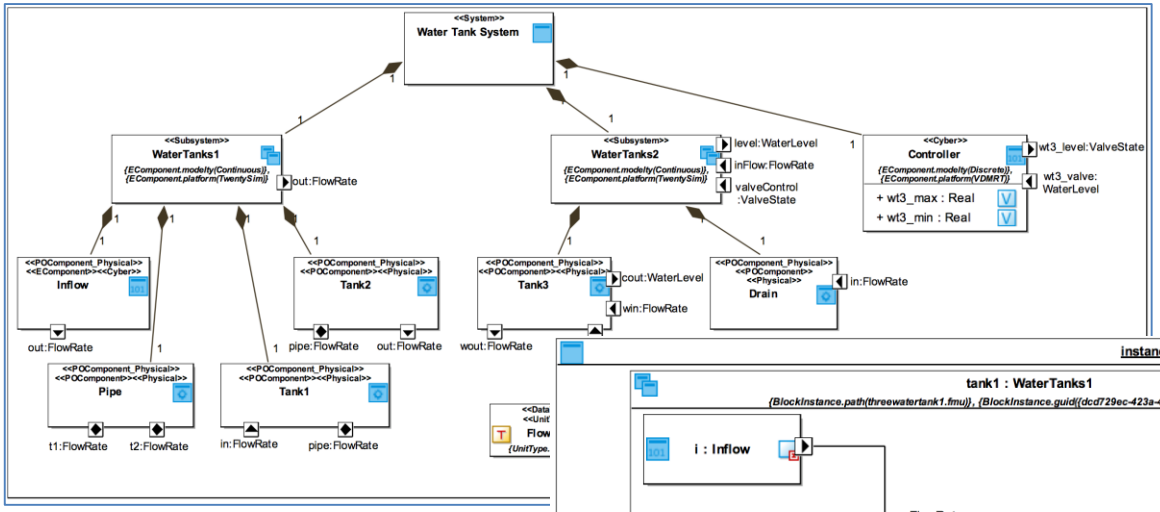


Underlying Unified Semantics



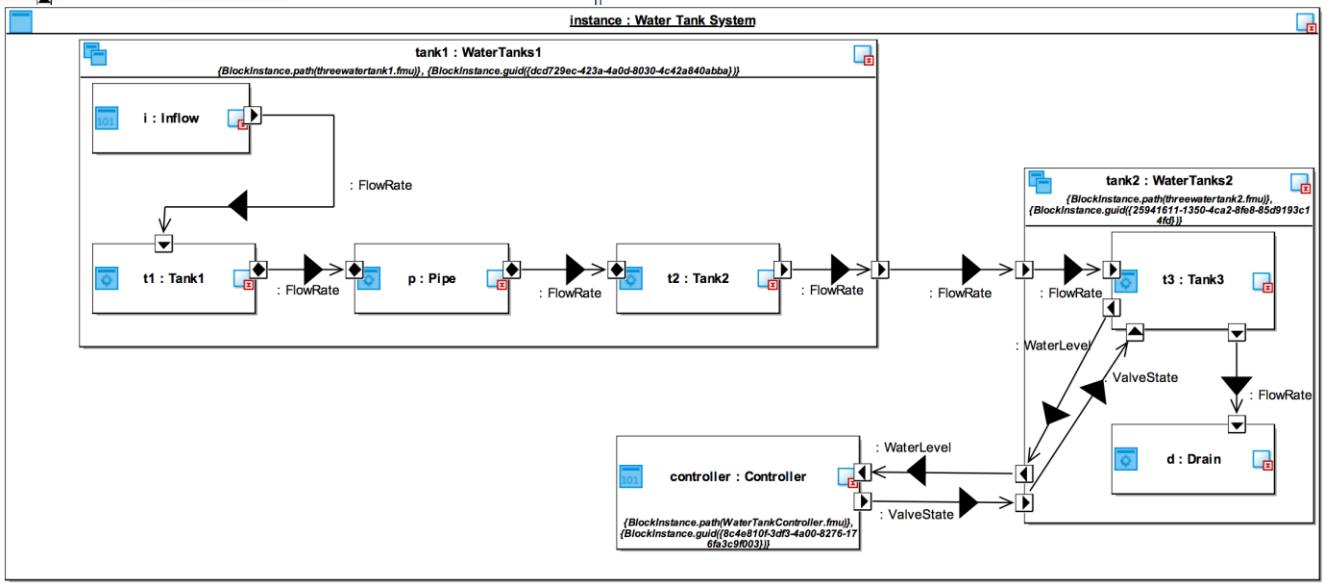
INTO-CPS SysML CPS Profile

- Three-tank Water Tank : INTO-CPS technology
  - Design architecture using INTO-CPS profile

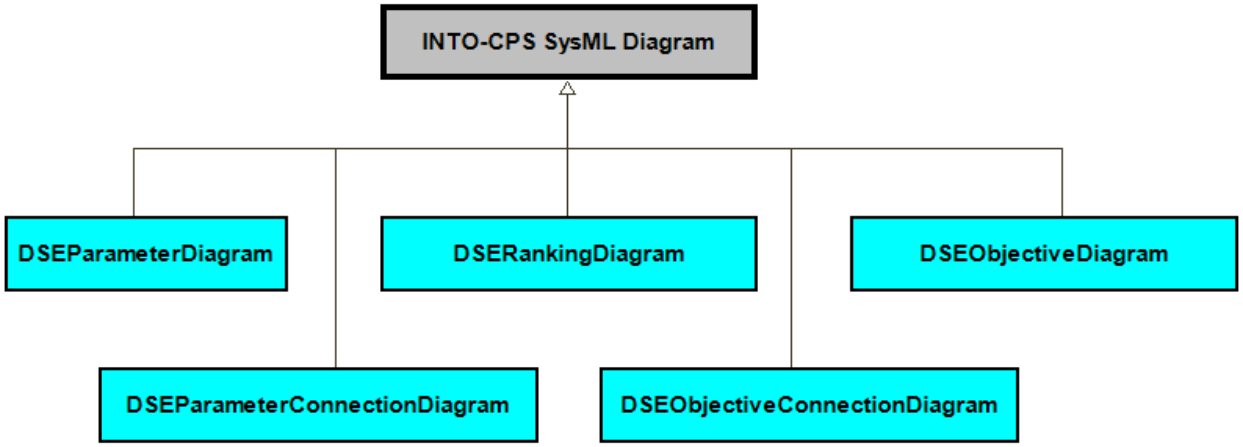


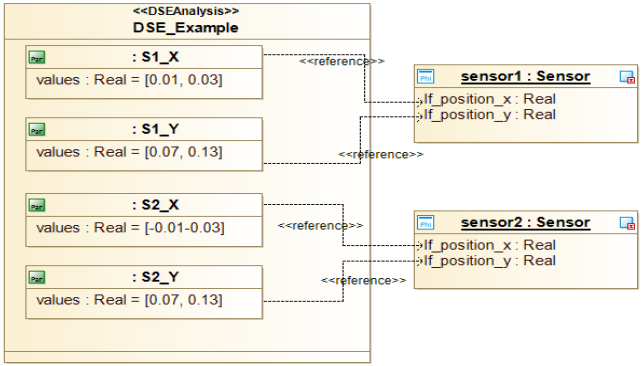
Architecture Diagram showing the connections between the different subsystems, and also their interfaces.

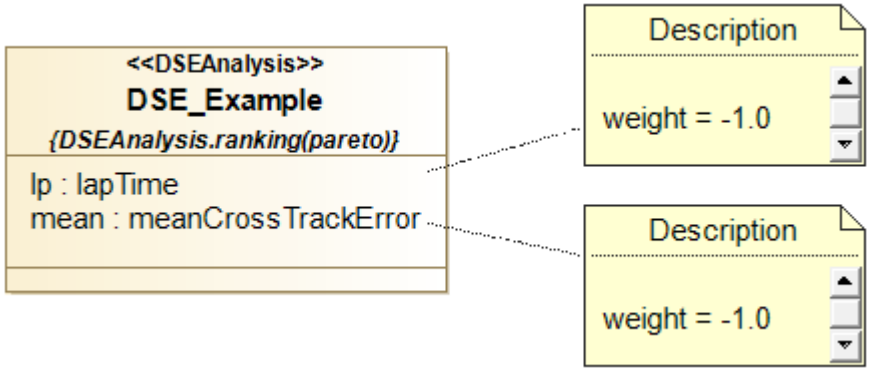
Architecture Diagram showing the connections between the different subsystems, and also their interfaces to the multi-model

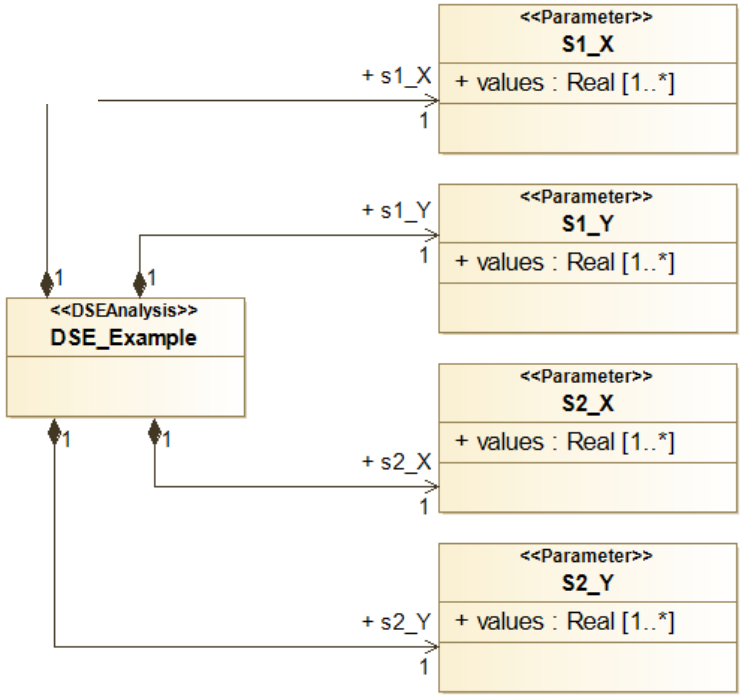


DSE Views in the SysML Profile

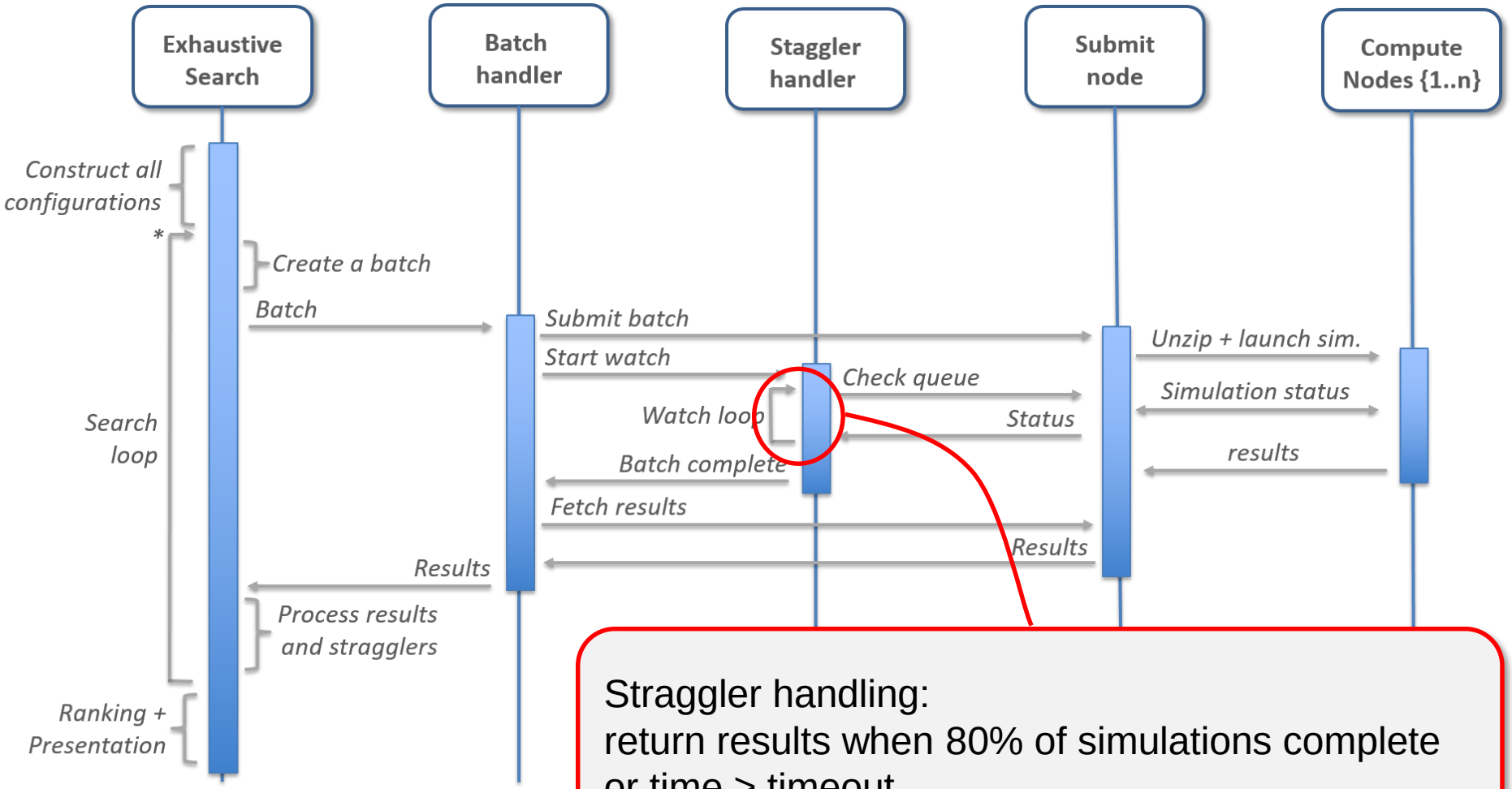








Cloud: Condor Exhaustive Search

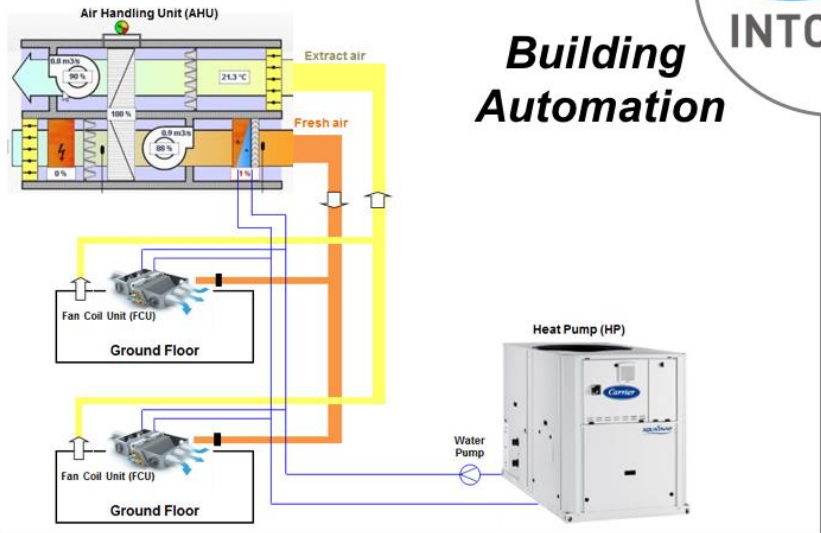


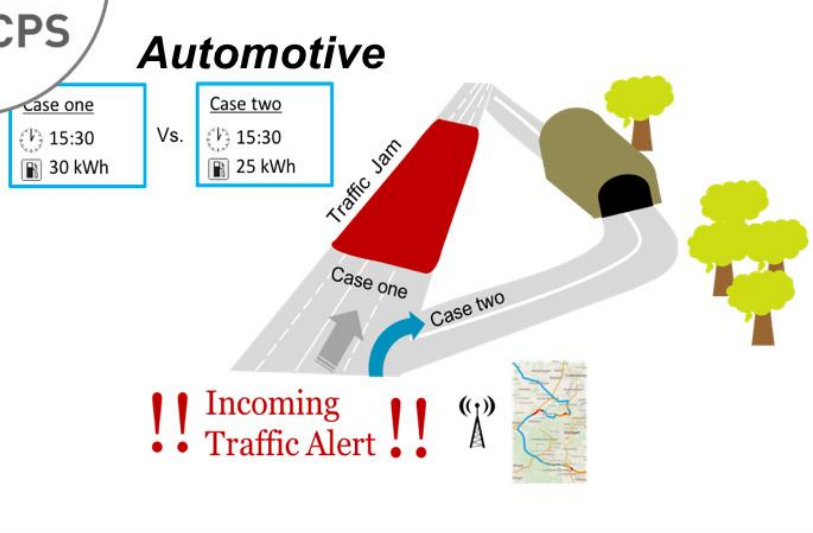
Straggler handling:
return results when 80% of simulations complete or time > timeout

Industrial Case Studies

INTO-CPS

INTO-CPS





Industrial Follower Group

AGCO, Denmark
Airbus, UK
Alcatel Lucent, Ireland
Almende, Netherlands
Alten, Sweden
Altran, UK
Andritz Hydro, Austria
Bachmann Electronic, Netherlands
Bakker-Sliedrecht, Netherlands
Bang&Olufsen, Denmark
Beia Consult, Romania
Beumer, Denmark
Bombardier, Germany
Bosch, Germany
Carrier, France
CCFE, UK
CeTIM, Netherlands
Chemring Technology, UK
Conpleks Innovation, Denmark
Continental, Romania
Critical Software, Portugal
Daimler, Germany
Danish Aviation, Denmark
Dassault Aviation, France
Delphi, Poland
Denso Corporation, Japan
Dredging International, Belgium
DSTL, UK
EDF, France
Enginsoft, Italy
European Space Agency, Netherlands
Fortiss, Germany
Goodrich, UK
Grundfos, Denmark
GN Resound, Denmark
HADATAP, Poland
Holonix, Italy
HMF, Denmark
Huisman Equipment, Netherlands
IBM, Israel
IBM, Finland
Ikergune, Spain
Inestec, Portugal
Irmato, Netherlands
ISMB, Italy
Jaguar, UK
John Deere, Germany
JNE Systech, Korea
MAN Diesel & Turbo, Denmark
MFAtech, UK
NII, Japan
Nupark Accelerace, Denmark
Odego, Germany
OFFIS, Germany
Omflow, UK
ONERA, France
Oticon, Denmark
Phillips MRI, Netherlands
PLM Consult, Denmark
Polar Electro, Switzerland
Postech, South Korea
Prime Solutions Group, USA
Projectglobe.com, UK
Rockwell-Collins, France
Rolls-Royce, UK
Saab, Sweden
Santer Reply, Italy
Seluxit, Denmark
Siemens, Sweden
Synelixis Solutions,Greece
Syntell, Sweden
TailSiT, Austria
Tecnalia, Spain
Terma, Denmark
Thalès R&T, Germany
TTTech, Austria
thyssenkrupp Marine Systems, Germany
UTC Aerospace, UK
Vesta System, France
West Consulting, Netherlands

In total: 80

The INTO-CPS Association Created

Purposes

- Maintain and develop further software that was created in the project
- Enable the widespread use of the software through open source licenses
- Promote its usage in academic and industrial settings
- Allow members to steer direction of development and incorporate software into products

INTO-CPS Association - Technologies

SysML profile
- Abstract modelling of CPS
- Builds on top of the Modelio tool (Open Source)
- Export of configurations for multi-models and DSE
- Traceability support

INTO-CPS application
- User Interface for setting up and executing Co-Simulation runs
- Integrates many elements of the INTO-CPS tool chain

INTO-CPS Traceability deamon
- Using OSLC to track updates from all tools

Co-Simulation Orchestration engine
- FMI 2.0 compliant master algorithm
- Supports Windows, Linux, Mac
- Parallelization and distribution support

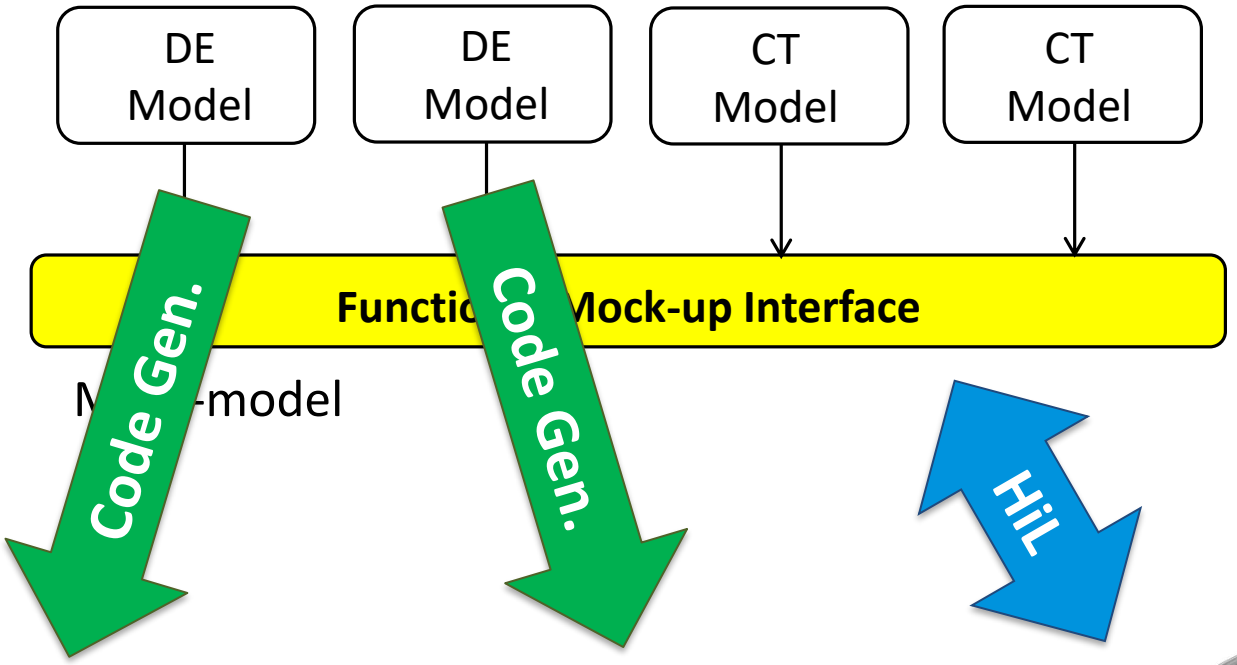
Design Space Exploration tools
- Algorithms for defining DSE runs and ranking the results
- Uses DSE configurations from Modelio
- Integrates well with the INTO-CPS application

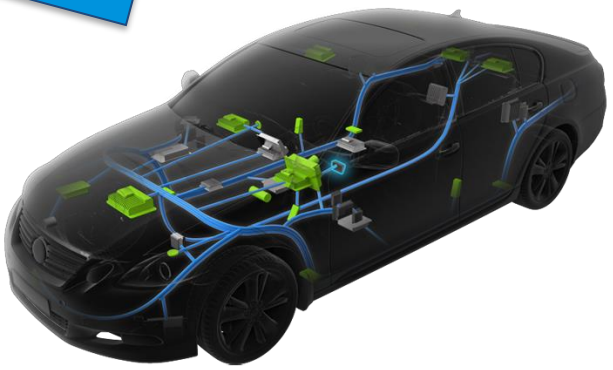
3D Animation FMU
- Unity support for FMU 3D animation

INTO-CPS Association - Licencing

General public	➢ View source-code ➢ Use software for non-commercial purpose
Associated members	In addition: use software for commercial purpose
Silver members	In addition: voting rights at General Assembly
Gold members	In addition: combine software with own software for commercial purpose, without the need to publish the code when

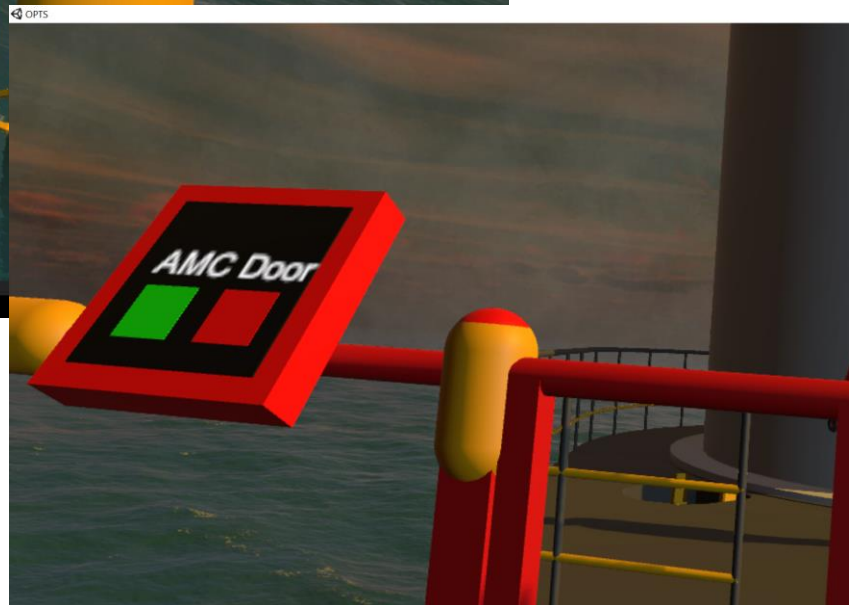
Hardware-in-the-Loop (HiL) and Code Generation





Oculus Rift and Simulation

INTO-CPS

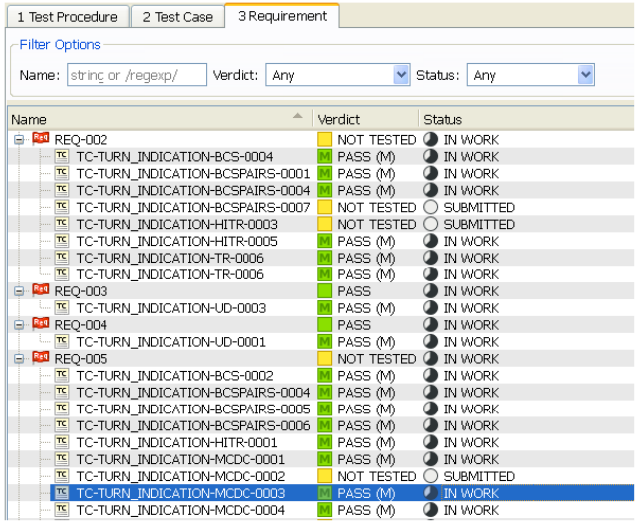


Horizon 2020 Programme

Seeing a Product before it exists

Test Automation

- Based on RT Tester tool suite
- Status:
  - Test sets generated from XMI import (from Modelio)
  - Test procedures are generated as FMUs, connected to Co-simulation
- Outlook:
  - Identify SuT in SysML profile, connect to Test Automation
  - Connect SysML requirements with LTL formulas



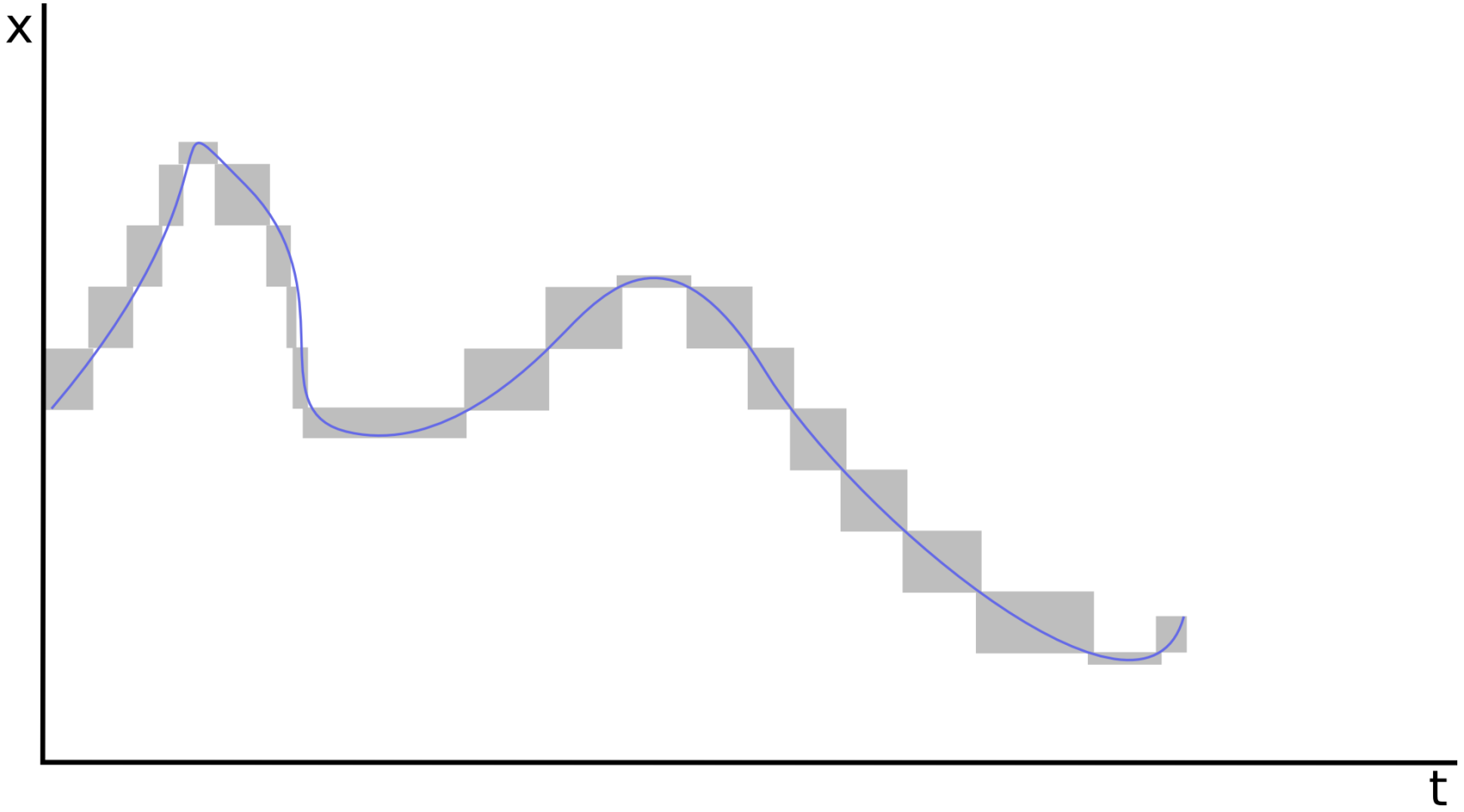
Model Checking Capabilities

- Now finding shorter counter examples.
- Integration of traceability.
- Provides abstractions for CT signals:
  - Interval-based abstraction
  - Simulation-based abstraction (new)
  - Gradient-based abstraction (new)

Simulation-based abstraction

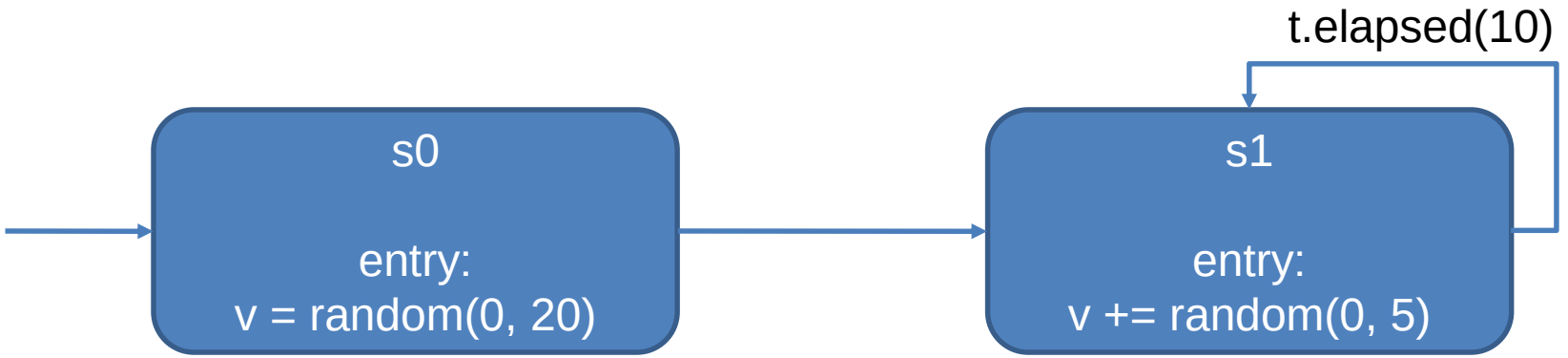
- During model checking a CT variable should behave similar to a previous concrete execution of the entire system.
- Extract signal flow from a test-run.
- Signal flow is abstracted by a succession of interval abstractions.
- Specify max allowed size of intervals to adjust granularity.

Abstraction by Interval Boxes



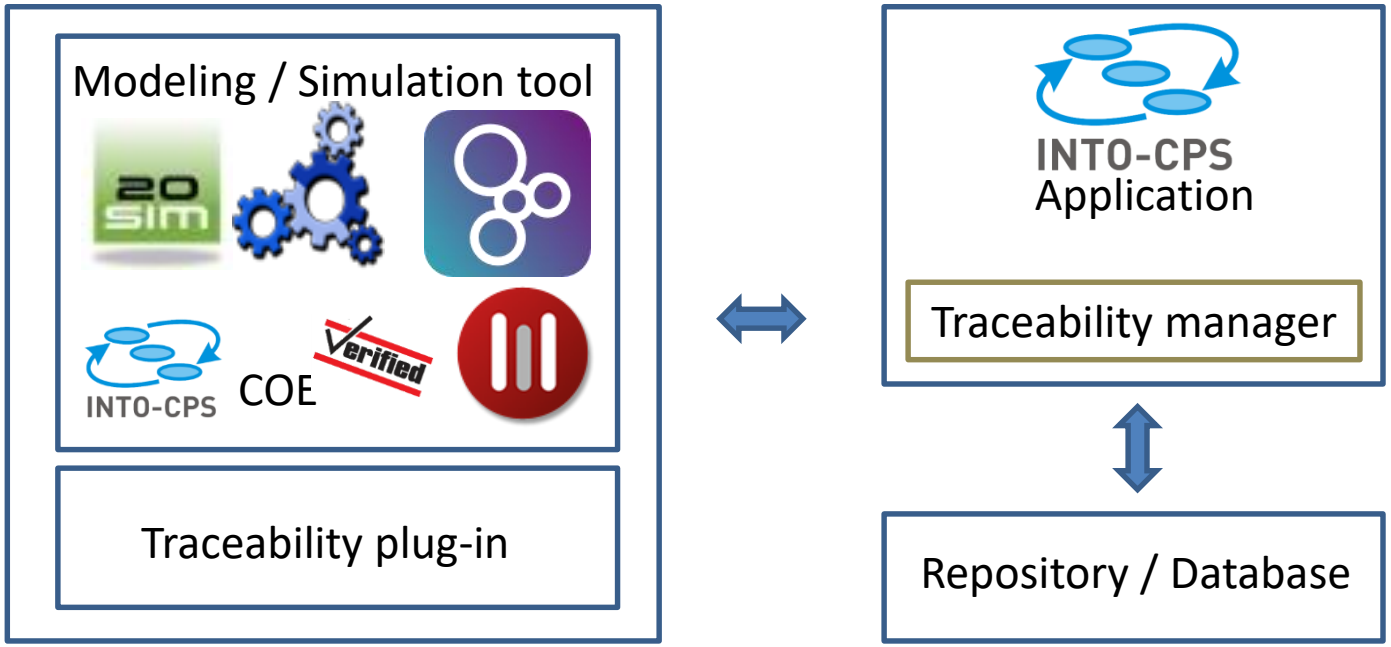
Gradient-based Abstraction

- Allow CT variable to increase/decrease only up to a certain amount per time.
- Implemented by a simple DE state machine.



Traceability & Provenance

- Goal: Ensure tracing between requirements, models, results, code
- Keep track of changes
- Use OSLC / Prov-N standards



Traceabilty – tool support

- Modelio
- OpenModelica
- Overture
- 20sim
- RT Tester
- INTO-CPS App

Requirements creation / linking with SysML blocks
Architecture modelling
ModelDescription.xml export
Configuration export

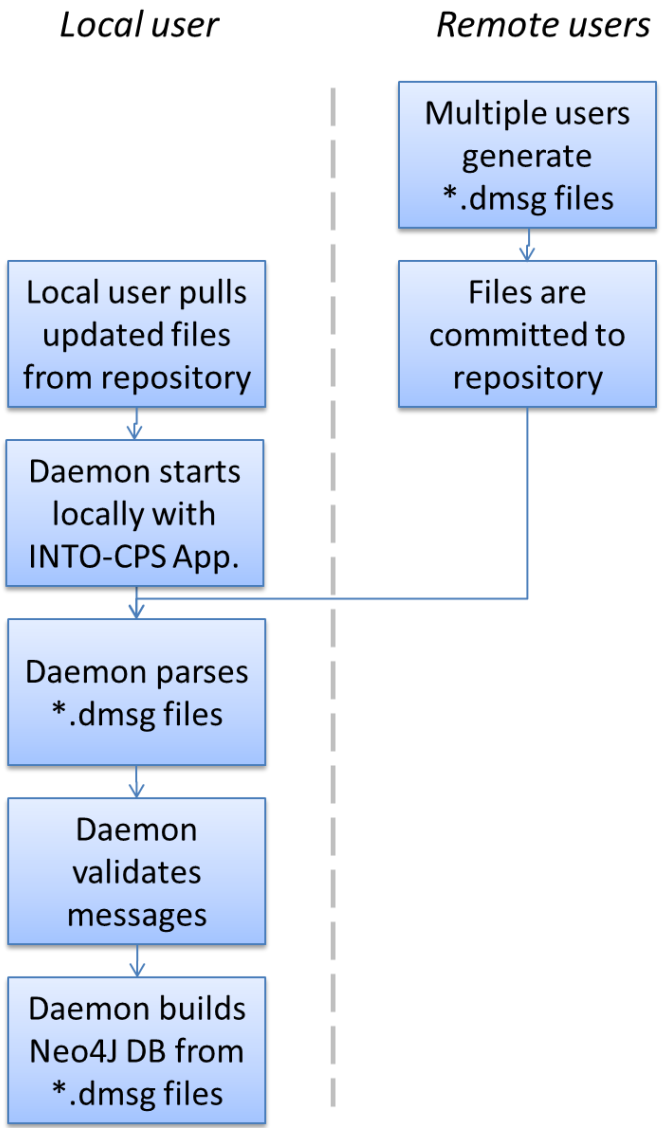
ModelDescription.xml import
Model creation / modification
FMU export

Define test model
Define test objectives
Run test
Define MC model
Define CT abstraction
Run MC query

Configuration to Multi-Model
Multi-model to COE configuration
Simulation

Traceability – multi-user support

- Each message is a single plain text file (*.dmsg)
- Messages are committed to common repository
- Each user can generate messages



Solutions for CPS Engineering Needs

- Enable collaboration across disciplines
  - Collaborative well-founded tool chain
- Keep development costs low
  - Lower need for physical tests by virtual co-simulation examination
- Keep time-to-market short
  - Enable concurrent engineering and gradual integration
- Explore the complex design space efficiently
  - Using Design Space Exploration
- Ensure tolerance against “nasty” faults
  - Experiment with what-if scenarios in a virtual setting
- Build up documentation for the working solution
  - Using combination of ad-hoc and automated tests
- Provide confidence to external stakeholders
  - Traceability between all project artefacts