CoHLA: Design Space Exploration and Co-simulation Made Easy

IEEE ICPS 2018, 17/05/2018

Thomas Nägele
t.nagele@cs.ru.nl

NWO Applied and Engineering Sciences

iCIS | Software Science
Radboud University

Contents

Introduction

The development of CPSs is hard . . .

- Multiple different disciplines
  - All having their own development methods and tools
- Concurrent development of components is difficult
  - Caused by component dependencies
- Impact analysis of design decisions is hard
  - Errors are detected (too) late

Introduction

Approach

- Model-based methodology to improve the development of CPSs
- Co-simulation of component models
  - Every component (model) as a simulation
  - Run all simulations together
  - And synchronise them
  - To simulate the system under development
- This approach provides early insight in the system design

Introduction

Contribution

- Simplify the creation of co-simulations
  - Using existing standards
- Early impact analysis of design decisions using Design Space Exploration (DSE)
- Collect performance metrics from the co-simulation

Introduction

Functional Mock-up Interface (FMI)

- Standard interface to support co-simulation
- Functional Mock-up Units (FMUs)
  - Contains model description, binaries and/or sources
  - Binaries can be simulated
- Co-simulation requires master algorithm
- Supported by a wide range of modelling tools

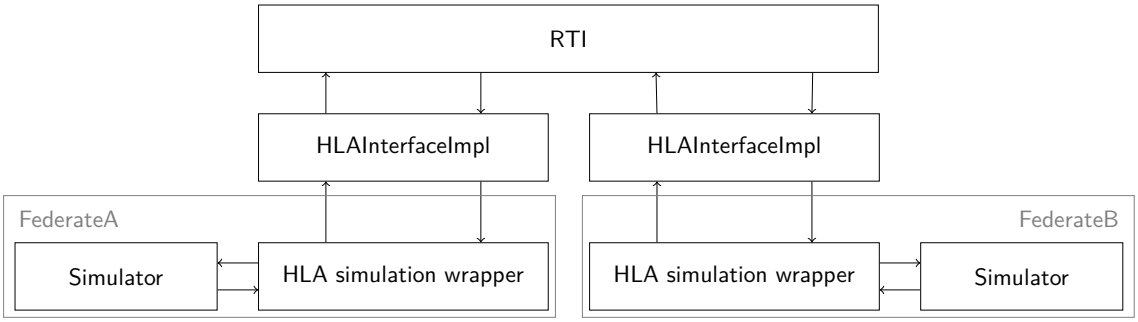
Parallel Object-Oriented Specification Language (POOSL)

- Discrete-time modelling language
- Suitable for modelling software architectures
- POOSL models simulated by the Rotalumis simulator

Introduction

High-Level Architecture (HLA)

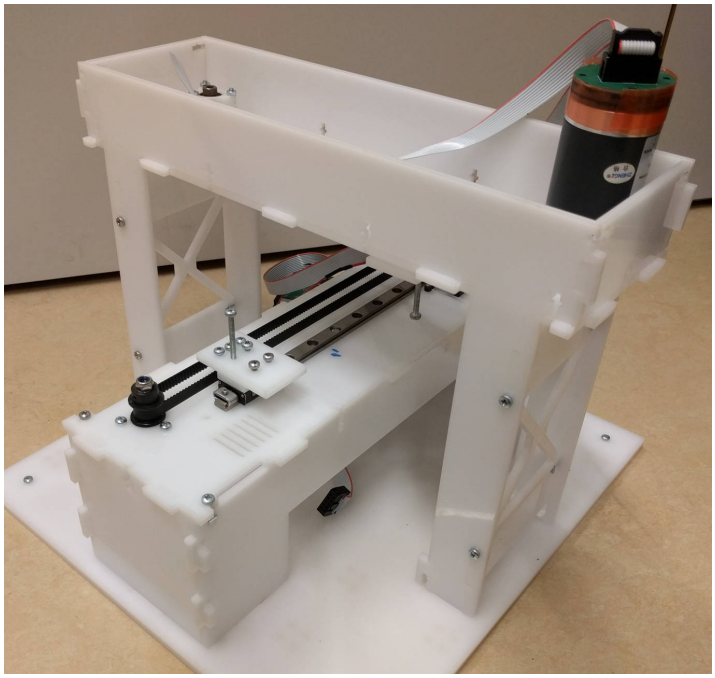
- Architecture specification for co-simulation
- Handles time and attribute synchronisation
- Federation containing Federates
- One central Run-Time Infrastructure (RTI)
- Simulators in co-simulation must comply to this standard
  - Usually requires wrappers to be implemented



CoHLA (Configuring HLA)

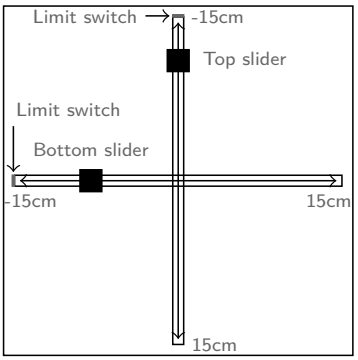
- Domain Specific Language (DSL) to construct co-simulations
- Developed with Xtext and Xtend (Eclipse)
- Co-simulation project is generated
  - Input: Co-simulation specification, simulation models
  - Output: CMake project, configuration files and run-script
- Currently supports OpenRTI
  - Simulation libraries
    - FMU
    - POOSL
  - Utility libraries
    - Collision detection and/or rendering
    - Logging
    - ...
- Enables quick changes during system design easily

Case study: Slider setup



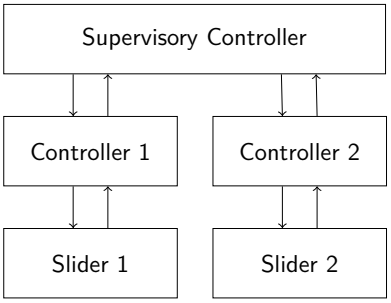
Case study: Slider setup

- Find a suitable initialisation procedure
  - Pre: Random starting position
  - Post: Slider position is known
  - Constraint: overshoot on limit switch should be within margins
- Performance metrics
  - Initialisation speed
  - Overshoot on limit switch
- Initialisation procedures
  - StrokeMode
  - FastMode
  - FixedMode
- All having their own parameter options



Case study: Slider setup

- Supervisory controller
  - Coordinates both sliders
  - Discrete-time
  - POOSL model
- Slider controller
  - Controller for a single slider motor
  - Discrete-time
  - 20-sim model
    - Modelling tool for mechatronic systems
- Slider
  - Physics model, including motor and limit switch
  - Continuous-time
  - 20-sim model



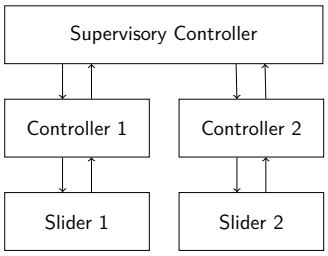
CoHLA: The basics

Simulation model definitions

```
1  FederateClass Axis {
2    TimePolicy RegulatedAndConstrained
3    Attributes {
4      Input Boolean enable
5      Output Real encoder
6      Output Boolean limit_switch
7      Input Real motor
8      InOutput Real position
9    }
10   Initialisables {
11     Initialisable Position_realInitial "position_real.initial" as
           Real
12   }
13   SimulatorType FMU
14   DefaultStep Time
15   DefaultModel "SliderAxis.fmu"
16 }
```

CoHLA: The basics

Co-simulation definition




```
Confederation SliderSetup {
  Instances {
    Instance bottomAxis as Axis
    Instance bottomController as Controller
    Instance sController as SupervisoryController
  }

  Connections {
    Connection { bottomController.limit_switch <- bottomAxis.limit_switch }
    Connection { bottomController.encoder <- bottomAxis.encoder }
    Connection { bottomAxis.motor <- bottomController.voltage }
    Connection { sController.bottomPosition <- bottomController.encoder_position }
    Connection { sController.bottomLimit <- bottomController.limit_switch }
    Connection { bottomAxis.enable <- sController.bottomEnable }
    Connection { bottomController.mode <- sController.bottomMode }
    Connection { bottomController.setpoint <- sController.bottomSetpoint }
    Connection { bottomController.stroketime <- sController.bottomStrokeTime }
  }
}
```


CoHLA: Extensions

Design Space Exploration (DSE)

DS Name	Attribute	Values
StrokeMode	Starting position (m)	0.15, 0.05, −0.05, −0.14
	Stroke time (s)	1.0, 1.5, 2.0, 2.5, 3.0
FastMode	Starting position (m)	0.15, 0.05, −0.05, −0.14
FixedMode	Starting position (m)	0.15, 0.05, −0.05, −0.14
	Movement speed (m/s)	0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20

```
DSEs {
  DSE strokeStartPositions {
    SweepMode Independent
    Situations : customMPs
    Set hlController.initMode : "0"
    Set bottomAxis.Position_realInitial :  "0.15", "0.05", "-0.05", "-0.14"
    Set hlController.initSpeed : "1.0", "1.5", "2.0", "2.5", "3.0"
  }
  DSE fastStartPositions {
    SweepMode Independent
    Situations : customMPs
    Set hlController.initMode : "1"
    Set bottomAxis.Position_realInitial :  "0.15", "0.05", "-0.05", "-0.14"
  }
  DSE fixedStartPositions {
    SweepMode Independent
    Situations : customMPs
    Set hlController.initMode : "2"
    Set bottomAxis.Position_realInitial : "0.15", "0.05", "-0.05", "-0.14"
    Set hlController.initSpeed : "0.02", "0.04", "0.06", "0.08", "0.1", "0.12", "0.14", "
        0.16", "0.18", "0.2"
  }
}
```

CoHLA: Extensions

Metrics

- Collect performance data from a DSE execution
- Several types of performance metrics
  - Error
  - EndValue
  - Min/max
  - Timer

```
1 MetricSets {
2   MetricSet Initialisation {
3     MeasureTime: 300.0
4     Metric InitialisationTime as Timer for hlController.
          initialised == true (EndCondition)
5     Metric MinBottomPosition as Minimum of bottomAxis.position
6   }
7 }
```


Case study: Results

- Co-simulation definition size
  - 6 files, 277 lines of code (LOC)
  - 4 model files + 5 3D model files
- Generated by CoHLA
  - 16 C++ source files, 1696 LOC
  - 10 C++ header files, 383 LOC
  - 1 CMake project file, 111 LOC
  - 1 FOM XML, 461 LOC
  - 1 run script, 886 LOC
  - 11 configuration files, 101 LOC
  - In total 40 files containing 3638 LOC
  - Even more important: **an executable co-simulation**

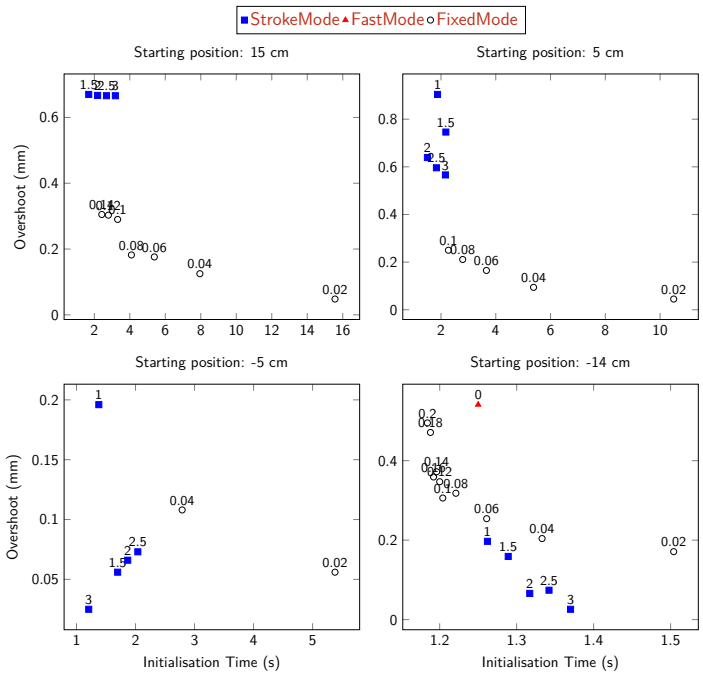
Case study: Results

Results

```
1  | ======= CONFIGURATION 0_0.15_1.0_customMPs =======
2  | ========             PARAMETERS             ========
3  | Situation                                : customMPs
4  | HlControllerHlController.initMode      : 0
5  | BottomAxisBottomAxis.position_realInitial: 0.15
6  | HlControllerHlController.initSpeed     : 1.0
7  | ========              RESULTS              ========
8  | InitialisationTime                       : 2.478010
9  | MinBottomPosition                        : -0.158773
10 | ================================================
11 |
12 | ======= CONFIGURATION 0_0.15_1.5_customMPs =======
13 | ========             PARAMETERS             ========
14 | Situation                                : customMPs
15 | HlControllerHlController.initMode      : 0
16 | BottomAxisBottomAxis.position_realInitial: 0.15
17 | HlControllerHlController.initSpeed     : 1.5
18 | ========              RESULTS              ========
19 | InitialisationTime                       : 1.685010
20 | MinBottomPosition                        : -0.150670
21 | ================================================
```


Case study: Results

Results



Conclusion

- CoHLA simplifies the creation of co-simulations of systems
- Easy Design Space Exploration

Current and future work

- Documenting and modelling CoHLA
  - Analysing timing behaviour
  - Provide insight in internals of (Co)HLA
- Implement additional CoHLA extensions
  - Distributed simulation execution
  - Requirement validation (automated)
- More case studies
  - Smart lighting system
  - Case studies at industrial partner

Questions

Thanks for your attention. Are there any questions?

iCIS | Software Science
Radboud University