Model-based System Design using a DSL for HLA Co-simulation

SEAA 2017 conference, August 31, 2017

Thomas Nägele
t.nagele@cs.ru.nl

NWO Applied and Engineering Sciences

iCIS | Software Science
Radboud University

Contents

Introduction

The DSL

Room Thermostat

Results

Conclusion

iCIS | Software Science
Radboud University

Introduction

Long-term goal

Develop a model-based methodology to improve the multi-disciplinary development process of Cyber-Physical Systems (CPSs).

- Virtual prototyping using models of both hardware and software
  - Current models often implicitly contain software components
  - Clear separation of software and hardware layers
- Concurrent development of software and hardware
- Allow flexible and scalable combination of models and real artefacts
  - HIL, SIL, ...
- Support industrial partner with early design decisions

Current goals

- Easy construction of co-simulation of different (types of) models
- Using a modular approach

Introduction

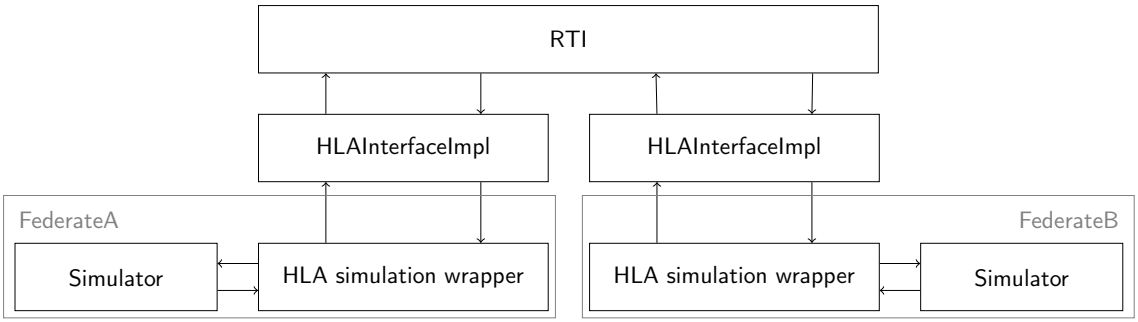
Technologies

- Functional Mock-up Interface (FMI)
  - Standard for model exchange and co-simulation
  - Supported by a wide range of modelling tools
  - Models can be exported to Functional Mock-up Units (FMUs)
- Parallel Object-Oriented Specification Language (POOSL)
  - Discrete-time modelling language
  - Very suitable for modelling software architectures
  - Maintained by TNO-ESI and TU/e
    - Close contact with maintainers

Introduction

Technologies

- High-Level Architecture (HLA)
  - Standard for co-simulation
  - Handles attribute and time synchronisation
  - Both commercial and public implementations available



Introduction

Implementation

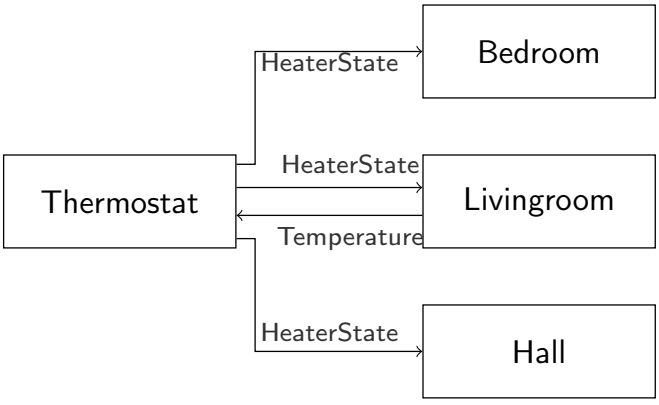
- Using the OpenRTI implementation of HLA
  - Open source C++ implementation
  - Multi-platform
- Implemented libraries to connect simulations to OpenRTI
  - Connector for FMI
  - Connector for POOSL
  - Logger library for CSV export
- Libraries decrease effort required for implementing a federate

The DSL

- Changes in system architecture require manual labour
- So we developed a Domain Specific Language (DSL)
- The DSL instance describes the co-simulation
  - Using Xtext and Xtend for Eclipse
  - Eclipse IDE for DSL instance definition
- Co-simulation project is generated automatically
  - Input: Co-simulation specification, simulation models
  - Output: Sources, makefile, FOM and run-script
- Co-simulation specification consists of 3 definitions
  1. RTI configuration
  2. Domain model definition
  3. Instance definition(s) & connections
- Allows easy coupling of models from POOSL, 20-sim, Modelica, ...
- Enables quick changes during system design

Room Thermostat – Co-simulation

- Room model containing a heater
  - Continuous-time 20-sim model
  - Exported to FMU
- Thermostat controller model
  - Discrete-time POOSL model
  - Simulated by Rotalumis



Room Thermostat – The DSL

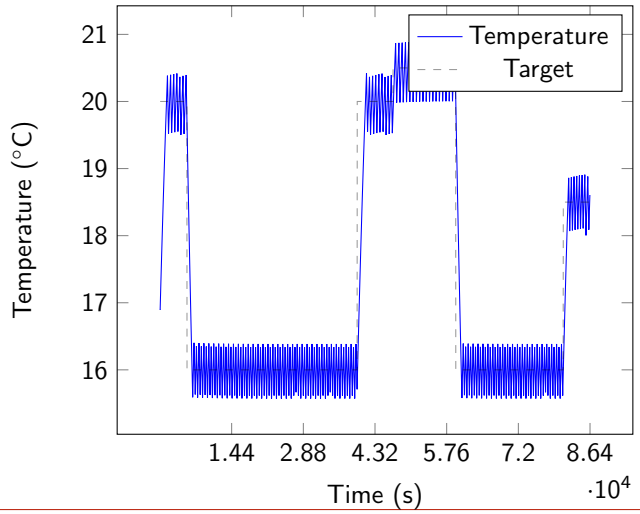
```
FederateClass Room {
  TimePolicy RegulatedAndConstrained
  Attributes {
    Input Boolean heaterState
    Output Real temperature
  }
  SimulatorType FMU
  DefaultStep Message
  DefaultModel "Room.fmu"
}
Confederation House1 {
  Instances {
    Instance Livingroom as Room
    Instance Thermostat as Thermostat
  }
  Connections {
    Connection { Livingroom.heaterState <- Thermostat.heaterState }
    Connection { Thermostat.temperature <- Livingroom.temperature }
}
```


Room Thermostat – Demo

iCIS | Software Science
Radboud University

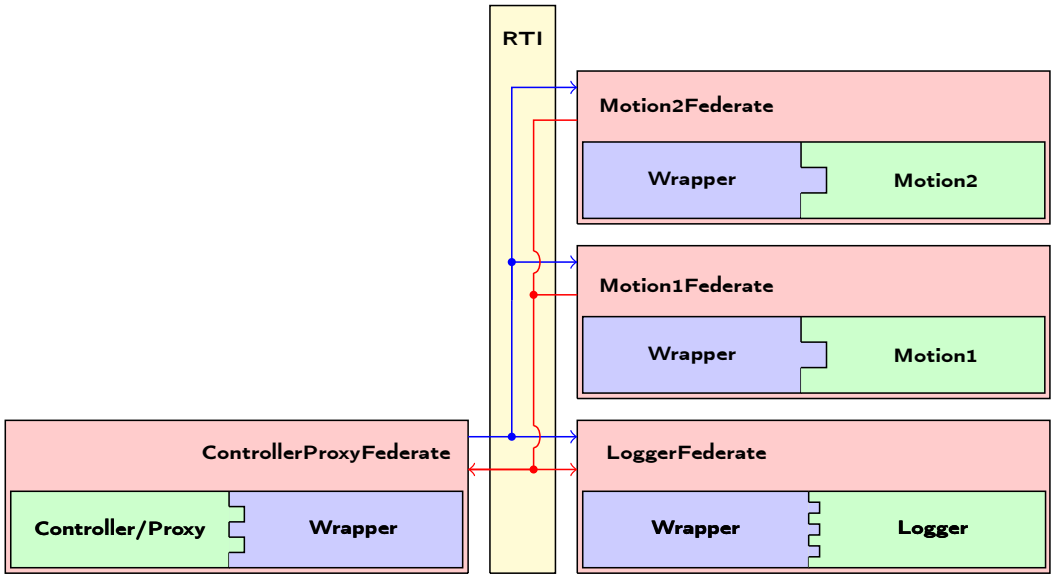
DSL improvements

- Initialisation variables for models
  - Useful for batch execution...
  - ...and design space exploration
- Sets of initialisation variables can be grouped into configurations
- Scenarios definitions to simulate external input



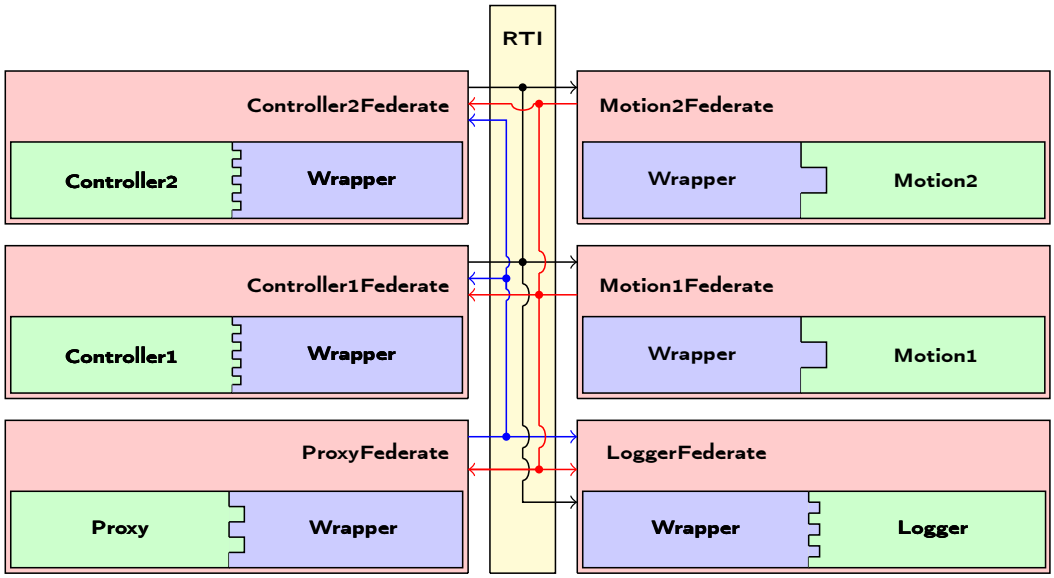
Results

- A method to apply changes to the system architecture quickly
  - An industrial example



Results

- A method to apply changes to the system architecture quickly
  - This change took less than an hour to complete!



Conclusion

Findings

- HLA provides a good basis for co-simulation
- Libraries simplify the creation of new federates
- The DSL decreases the effort required to develop the co-simulation
  - Easy changes in system architecture
  - Very suitable for design-space exploration
  - Can be extended rather easily

Future work

- Develop a methodology for system design based on the DSL
  - Including fault injection and DSE
  - Requirement validation
- More case studies using our approach
- Investigate scalability of the method
- Experiment with distributed simulations

iCIS | Software Science
Radboud University

Questions

Thank you for you attention! Are there any questions?