The INTO-CPS Co-Simulation Orchestration Engine – Experiences with FMI 2.0 and proposed extensions

Christian König, TWT GmbH, Germany
FMI User meeting / Prague

15/05/2017

INTO-CPS

AARHUS UNIVERSITY
UNITED TECHNOLOGIES
Newcastle University
MODELIO SOFT
The University of York
Verified
AGRI INTELLIGENCE
Linköping University
TWT GmbH
Science & Innovation
CLEARSY
SYSTEM ENGINEERING
A New Toolchain for CPS Design

- Requirements
- Heterogeneous Systems Models
- Code / Hardware
  - HiL / SiL Simulation
  - MiL Co-Simulation
  - Feedback
  - Design Space Exploration
  - Test Automation

SysML - FMI Model Generation
FMI related

Strong Traceability
Configuration Management
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)
- GUI 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
Performance

• `getMaxStepSize()` Proposed by D. Broman et al (Determinate composition of FMUs for Co-Simulation, 2013)
  – Required to improve simulation speed for FMUs that don’t support roll-back (set previous step by `fmi2GetFMUstate / fmi2SetFMUstate`)
  – Tools that implement rollback: Dymola, 20-sim
  – `getMaxStepSize()`: Overture

• pointer references were found to improve performance instead of get/set
  → any other experience?
Parallelization

- Parallelization (here using Scala) showed varying performance enhancement:
  - Parallel execution of getFMUxxx / setFMUxxx / doStep
  - Initial results show at 15 - 30% performance increase for a standard Co-simulation model
  - thread synchronization costs time
  - Performance depends strongly on models → logic needed to sort execution of FMUs for optimal performance

- Distributed Co-Simulation
  - Allows using mixed 32/64-bit FMUs
Cross check / build

• Only single FMU simulations are checked, no Co-Simulation

• Suggestion: at least two FMUs should be checked for FMI-CoSimulation
  – All FMUs from same tool / vendor
  – Different tools / vendors

• Compilation information is missing for source FMUs

• INTO-CPS has created a cross-compilation service for all target architectures (Mac, Linux, Windows)
  ➔ https://sweng.au.dk/fmubuilder/
Additional ressources of interest

• Some FMUs generate additional analytical data
  – Internal timed state transitions, can be used for model checking
  – Tools from Verified Systems generate information on model validity

→ Standardized description of internal FMU behaviour is desirable for post-analysis
Discrete systems

• Network protocols can be simulated by combination of strings and booleans

• However, scalability is poor, delays are caused

• Ether model: https://github.com/into-cps/case-study_ether

• Guidelines for modelling of discrete systems would be very helpful

• Composite types (e.g. lists) would be desirable for discrete systems, such as controllers
Deliverables and Outreach

- FMI related Deliverables available on website
  
  http://into-cps.au.dk

  - D4.1d – Design of the INTO-CPS plattform
  - D4.2a – User manual
  - D4.2b – Integration of simulators
  - D4.2c – SysML Contracts
  - D2.1d & D2.2d – Foundations for FMI Co-Modelling

- Industry & Academic Follower Group to be involved with project progress
The INTO-CPS Co-Simulation Orchestration Engine – Experiences with FMI 2.0 and proposed extensions
Christian König, TWT GmbH, Germany
FMI User meeting / Prague
15/05/2017