The Mars-Rover Case Study Modelled Using INTO-CPS

Sergio Feo-Arenis, Marcel Verhoef and Peter-Gorm Larsen

15/09/2017
Goals of the Study

• Understand the effort required to migrate existing co-simulation models from DESTECS into INTO-CPS.
• Explore the advantages and limitations of using INTO-CPS for co-simulation workflows. In particular, the FMU/FMI paradigm.
INTO-CPS

- Configure / launch
  - Co-simulation
  - Design Space Exploration
  - Model Checking
  - Test Automation
- View Results
- Traceability

Modelio

Overture

20-sim

OpenModelica

RT-Tester

FMU

FMU

FMU

FMU

FMU

FMU

FMU

COE (Co-simulation Orchestration Engine)

Result

Result

Result

Traceability Daemon
The Mars Rover Case Study

- **Aim:** to develop a co-model to evaluate controller strategies through simulation
- **Proprietary ESA Model of vehicle dynamics**
- **Wish to evaluate different controller proposals without disclosing the model**
Co-Simulation Model

- Control strategies
- Mode switching
- Safeguard mechanism

VDM-RT

- Simplified geometry
- Environment model
- Contact model

20-Sim

Discrete Controller Model

Continuous-Time Model
Overture FMI Extension

- Automatic inclusion of FMU library
- Automatic import of model descriptions from Modelio
- Requires modifying models:
  1. Add `HardwareInterface` class containing exchange ports, their types (bool, int, real and string) and exchange directions (input / output)
  2. Adapt model to read and write from the hardware interface using dedicated methods
- Automatic generation of FMU
  - Tool wrapper
  - Source code
Overture FMI Extension

FMU data exchange functionality
class HardwareInterface
values
   -- @ interface: type = parameter;
   public v : RealPort = new RealPort(1.0);

instance variables
   -- @ interface: type = input;
   public distanceTravelled : RealPort := new RealPort(0.0);
   -- @ interface: type = output;
   public setAngle : RealPort := new RealPort(0.0);

end HardwareInterface
20-Sim FMI Extension

- Declaration of external variables specifying type and direction
- No need to use dedicated functions for reading or writing
- Support code generation (20-Sim 4C) and code-based FMU (experimental)
- Support tool wrapper FMU generation

```plaintext
externals
   // External variables
   real global export Actual_Steering_1;
   real global export Actual_Steering_2;
   real global export Actual_Steering_3;
   ...
   real global import steering_1;
   real global import steering_2;
   real global import steering_3;
   ...

equations
   ...
```
20-Sim Model
Results

- Performed model adaptation for FMI generation
- Created an INTO-CPS co-model from generated FMUs
- Effort required:
  - ~20 work hours for model modification: learning curve, addition of FMI-specific constructs, configuration of the co-model, debugging
  - Involvement of tool vendors for the resolution of issues with software components (20-sim FMU generation, contact model update)
- Able to reproduce the simulation outcome of the INTO-CPS setup
Results
Challenges Encountered

- Model compatibility issues, update required for current tool versions
  - Interface definitions
  - VDM reserved keywords
  - Upgraded contact model
- Experimental status of the tools involved.
- No support for interrupt-like events in co-simulation mode. Requires additional effort.
- No support for vector types. Requires either:
  a. Encoding/Decoding via strings.
  b. Naming conventions plus additional tooling layer.
Advantages and Outlook

- Management of intellectual property
- Possibility of distributed model construction
- Possibility of distributed, heterogeneous simulation
- Possibility of “system of systems” mission analysis
- Possibility of early validation of on-board software
  - Currently working on code generation from discrete models (vdm2c)
  - Allow integration into co-models as well as deployment into hardware targets using TASTE