A Generic FMU Interface for Modelica

Wuzhu Chen¹  Michaela Huhn¹  Peter Fritzson²

¹Institut für Informatik, Technische Universität Clausthal, Germany
{Wuzhu.Chen | Michaela.Huhn}@tu-clausthal.de

²Department of Computer and Information Science, Linköping University, Sweden
Peter.Fritzson@liu.se
Overview

1. Introduction to Modelica & FMI
2. Approaches for FMU Import
3. Prototype Implementation
4. Case Study
5. Conclusion
Introduction 1

- **Model Exchange in Modelica**
  - Modelica is an EOOL for modeling and simulation
  - A bunch of modeling and simulation tools based on Modelica
  - Generally no problem for model exchange btw. these tools
Introduction 2

- **Functional Mock-up Interface (FMI) 1.0 for Model Exchange**
  - Specification of a C interface for models (FMUs)
  - An FMU instance presents the model attributes and behavior
  - FMUs are distributed in a compressed form
  - FMUs may be exchanged between any tools for variant purposes
Introduction 3

- **Enhance Model Reusability and Interoperability in Modelica**
  - Functionality for FMU export to non-Modelica-based tools
  - Functionality to import generated FMUs

![Diagram showing FMU exchange between Modelica Tools and Non-Modelica-Based Tools](attachment:image.png)
Overview

1. Introduction to Modelica & FMI
2. Approaches for FMU Import
3. Prototype Implementation
4. Case Study
5. Conclusion
Approaches for FMU Import 1

- **Stand-alone FMU Import**
  - Decompression of the archived .fmu file
  - Parsing model description XML file
  - Wrapper around interface functions
  - Connection with solver
  - Result analysis and presentation

- **FMU Import in Modelica**
  - Decompression of the archived .fmu file
  - Parsing model description XML file
  - Mapping interface functions onto Modelica external functions
  - Mapping FMI structures onto Modelica constructs
  - Integration of model attributes and behaviors
Approaches for FMU Import 2

- Workflow of FMU Import in Modelica
Overview

1. Introduction to Modelica & FMI
2. Approaches for FMU Import
3. Prototype Implementation
4. Case Study
5. Conclusion
Prototype Implementation 1

1. Decompression of FMUs
   - 7-zip is used to decompress FMUs
   - Decompressed files stored in a sub-directory in OpenModelica
   - Temporary files may be deleted after code generation

2. Parsing modelDescription.xml
   - Model variables and their attributes are stored in this xml file
   - Open-source library Expat is used
   - Result is a tree-like structure with nodes attached on it
   - Variable attributes can be queried by traversing the structure
   - Validation of the xml file not yet implemented
Prototype Implementation 2

3. Generic Interface Generation

- FMI interface functions in a dynamic link library (dll)
- Load the library via information from the xml file
- Wrapper functions are then created around FMI functions
- Some extra helper functions
- FMI interface is represented as Modelica external function constructs
- Code generation from a template file
Prototype Implementation 2

4. Modelica Models Generation

- Calling sequence guaranteed by Modelica *algorithm* constructs
- Specific data-flows ensured by the restricted class *block*
- Internal variables defined based on the information from the xml file
- The FMU block body generated by incorporating variable information and interact with the model via interface functions
Overview

1. Introduction to Modelica & FMI
2. Approaches for FMU Import
3. Prototype Implementation
4. Case Study
5. Conclusion
Case Study 1: Heat Transfer Equation

- **Imported CPU Cooling Model (Dymola 7.4)**
  - Continuous dynamic system
  - FMU import functionality of the prototype from other simulator in OpenModelica
Case Study 1: Heat Transfer Equation

- Original Modelica CPU Cooling Model (Dymola)
Case Study 2: Hybrid DAE System

- **Bouncing Ball Model (fmusdk)**
  - Hybrid dynamic system
  - Testing FMU import functionality of the prototype
Case Study 2: Hybrid DAE System

- **Bouncing Ball Model (fmusdk)**
  - Testing capability of multiple instances of an FMU
Case Study 2: Hybrid DAE System

- **Bouncing Ball Model (fmusdk)**
  - Testing capability of native-connection btw. the imported FMU and pure Modelica model
Overview

1. Introduction to Modelica & FMI
2. Approaches for FMU Import
3. Prototype Implementation
4. Case Study
5. Conclusion
Conclusion

- We realized
  - FMU Import
  - multiple FMU instances
  - native-connections between imported FMU and pure Modelica model
- The interface is vendor-neutral and open-source
- Automatic code-generation fully compliant with Modelica language specification is achieved as well
Thanks for your attention!
Questions are welcomed.