Linking FMI-based Components with the Ptolemy Discrete Event System

7th MODPROD Workshop on Model-Based Product Development

Edmund Widl, Wolfgang Müller

Austrian Institute of Technology – Complex Energy Systems

edmund.widl@ait.ac.at, wolfgang.mueller.fl@ait.ac.at
Motivation: Complex cyber-physical energy systems

- Modeling/simulation requires the **combination** of various domains:
  - **physical world**: continuous models, e.g. energy generation, transport, distribution, consumption, …
  - **information technology**: discrete models, e.g. controllers, communication infrastructure, software, …
  - **roles/individual behavior**: game theory models, e.g. agents acting on behalf of a customer, market players, …
  - **aggregate/stochastic components**: statistical models, e.g. weather, macro-view of many individual components

- **Validated tools** available for all sub-domains
  - How to combine the expertise of all these tools?
A possible approach

- **Problems:**
  - *Reimplementation* of specialized tools **not viable**
    - re-use existing expertise
  - *Traditional* simulation approaches **impractical**
    - system size
    - amount of events
      - e.g. Modelica: each event, even in a small sub-component, triggers a costly global event handling
A possible approach

- **Problems:**
  - *Reimplementation* of specialized tools **not viable**
    - re-use existing expertise
  - *Traditional* simulation approaches **impractical**
    - system size
    - amount of events
      - e.g. Modelica: each event, even in a small sub-component, triggers a costly global event handling

- A possible solution: **Co-simulation**
  - *Discrete event-based* master algorithm
  - *Continuous time-driven* components as “plug-ins”
  - *Hierarchical* modeling approach
    - handle events where they happen, propagate effects when necessary

- **Work in progress!**
Discrete event-based simulation using Ptolemy II

- **Open source tool**
  - *generic framework* for studying **concurrent processes**
  - *implements* many simulation domains
    - used here: **Discrete Event (DE) domain**

- **Concurrent processes**
  - represented by so-called **actors**
  - actor implementations have to obey certain guidelines (interface & behavior)
    - referred to as *abstract semantics*
  - interaction of actors (communication and order of execution) is governed by so-called **directors**
    - directors implement domains (*models of computation*)
  - enables **hierarchical models**
Use FMI for continuous time-based components

- A **tool independent** standard for model exchange and **co-simulation**
  - supported by various modeling environment focusing on **continuous time-based** simulation
  - ideal for a **multi-purpose** co-simulation environment
Use FMI for continuous time-based components

- A tool independent standard for model exchange and **co-simulation**
  - supported by various modeling environment focusing on *continuous time-based* simulation
  - ideal for a *multi-purpose* co-simulation environment

- **Coupling** to Ptolemy II via the **FMI++ library**
  - C++ library implementing *generic high-level functionalities* for FMUs
    - on top of FMI ME
    - self-integrating FMUs
    - advanced event-handling
  - Allows JAVA-bindings via **SWIG**
  - **Work in progress!**
Event handling via the FMI++ library

- Discrete **time-step** simulation
  - used e.g. in real-time simulation
  - use *small* step size, handle events *as they occur*
- Discrete **event-based** simulation
  - *predict* occurrence of *next* internal event
  - *react* to external events
- FMI++ library: **lookahead predictions**
  - Work in progress!
Event handling via the FMI++ library

- Discrete **time-step** simulation
  - used e.g. in real-time simulation
  - use *small* step size, handle events *as they occur*
- Discrete **event-based** simulation
  - *predict* occurrence of *next* internal event
  - *react* to external events
- FMI++ library: **lookahead predictions**
  - Work in progress!
Event handling via the FMI++ library

- Discrete **time-step** simulation
  - used e.g. in real-time simulation
  - use *small* step size, handle events *as they occur*
- Discrete **event-based** simulation
  - *predict* occurrence of *next* internal event
  - *react* to external events
- FMI++ library: **lookahead predictions**
  - Work in progress!
Event handling via the FMI++ library

- Discrete **time-step** simulation
  - used e.g. in real-time simulation
  - use *small* step size, handle events *as they occur*
- Discrete **event-based** simulation
  - *predict* occurrence of *next* internal event
  - *react* to external events
- FMI++ library: **lookahead predictions**
  - Work in progress!
Mapping of Ptolemy II’s Semantics to FMI++

FMI ME specification

- initialize
- instantiate
- setContinuousStates
- setTime
- completedIntegratorStep
- … and many. many more
Mapping of Ptolemy II’s Semantics to FMI++

FMI ME specification
- initialize
- instantiate
- setContinuousStates
- setTime
- completedIntegratorStep
- … and many. many more

FMI++: IncrementalFMU class
- initialize
- synchronize
- checkForEvent
- handleEvent
Mapping of Ptolemy II’s Semantics to FMI++

FMI ME specification
- initialize
- instantiate
- setContinuousStates
- setTime
- completedIntegratorStep
- ... and many, many more

FMI++: IncrementalFMU class
- initialize
- synchronize
- checkForEvent
- handleEvent

Ptolemy II: FMU-ME actor
- initialize
- prefire
- fire
- postfire
- wrapup

high-level wrapper

Ptolemy abstract semantics
Simple Example of an FMU in Ptolemy II’s DE Domain
Simple Example of an FMU in Ptolemy II’s DE Domain
Simple Example of an FMU in Ptolemy II’s DE Domain
Conclusions and Outlook

- **Investigation** of *co-simulation paradigm*
  - discrete event-based master algorithm, continuous time-driven plug-ins
  - hierarchical modeling
  - re-use of validated concepts

- **Work in progress**, but first *promising* results
  - concept for coupling FMUs (ME) to discrete event-based simulation

- Make code available to **open-source** community in the near future
  - *FMI++ library*
  - Ptolemy II *FMU-ME actor*
  - ...

- **Contributions and suggestions are welcome!**
Backup: Functional Mock-up Unit (FMU)

- A zip-file containing the XML model metadata file and dynamic libraries, according to the FMI-standard for model exchange
- Can have continuous and discrete states, supports event handling
- A solver (integrator) has to be provided externally
Backup: Example result