FMI-based Co-Simulation of Hybrid Closed-loop Control System Models

Edmund Widl, Florian Judex, Katharina Eder

Austrian Institute of Technology, Vienna, Austria
{edmund.widl, florian.judex, katharina.eder}@ait.ac.at

Peter Palensky

Delft University of Technology, Delft, The Netherlands
p.palensky@tudelft.nl

***Abstract*—In recent years, co-simulation and model exchange approaches have become ever more popular within the context of model-based design and analysis. This trend has led to the development of the Functional Mock-up Interface (FMI) specification—the de facto standard for model exchange and co-simulation—and a growing number of FMI-compliant tools.**

This paper addresses the FMI-based co-simulation of hybrid models representing closed-loop control systems, where a continuous time-based plant model is connected to a discrete event-driven controller model. The semantics of execution and data flow of such models are discussed and demonstrated with the help of a model that has been inspired by real-world applications. An example illustrates that popular proprietary simulation environments are not necessarily able to properly capture the semantics of these models. Furthermore, it is shown how existing concepts and tools can be successfully applied to implement such simulations properly.

***Index Terms*—Modeling, simulation, simulation software, closed-loop control, Functional Mock-up Interface.**

List of used Abbreviations

API	Application Programming Interface
CS	Co-Simulation
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
ME	Model Exchange
ODE	Ordinary Differential Equation

I. Introduction

The requirements for modern controls are constantly rising. Trans-disciplinary processes, for instance encountered in integrated energy systems, meet ever more complex requirements for controls. Designing controls for such situations requires detailed, multi-domain models of the systems under control. Modeling the entire system of systems in one universal language, with one tool, leads to simplifications that can exclude important properties and dynamic dependencies [1]. However, specialized tools have validated libraries, optimal solvers, and a language that perfectly fits the problem domain. Therefore, combining such specialized simulation tools in a co-simulation approach solves several problems in designing and analyzing modern controls.

Once it is understood how such hybrid models can be easily created and put to use, properties like their dynamic stability limits, scalability issues and other complex phenomena can be investigated. However, the associated challenges are numerous, including

- interfacing the simulation tools/models in a standardized way,
- understanding numeric phenomena when combining two or more tools/models, and
- efficiently connecting discrete controller models with continuous plant models.

The most promising approaches to do so are currently based on the *Functional Mock-up Interface* (FMI) [2] specification, which can be considered the de facto standard for co-simulation and model-exchange. With the development pushed by both industry and academia, a growing number of simulation software providers (commercial and community-driven) has adopted and integrated FMI into their products. This allows well-established and specialized simulation tools from a wide spectrum of application domains to be coupled and re-used with only small adaptations in contrast to monolithic approaches [3].

However, due to the FMI specification's strong focus on time-continuous (physical) systems, the inclusion of time-discrete modeling aspects still remains in its early stages. Hence, the proper handling of hybrid systems combining discrete and continuous models is a delicate topic. This is also true for control applications where discrete controllers act on physical plants. The closed-loop nature of such control systems poses a fundamental challenge for FMI-based co-simulation, where algebraic loops are only poorly understood from a formal perspective.

The remainder of this article is structured as follows. In Section II a short overview on FMI-based simulation in general is given. Section III presents a detailed description of the semantics of execution and data flow of hybrid FMI-based closed-loop control system models. Section IV introduces a simple reference model inspired by real-world applications, which is used for a comparative study in Section V. Finally, Section VI summarizes the findings and closes with an outlook.

II. State of the Art

The FMI specification provides simple and generic but powerful interfaces that can be adopted for a large range of simulation tools. A simulation component compliant to the

978-1-4673-7178-0/15/$31.00

FMI specification is called a *Functional Mock-up Unit* (FMU). Version 2.0 of the FMI specification defines two specific types of interfaces:

- FMI for Co-Simulation (CS) specifies stand-alone black-box simulation components that can be executed independently from any simulation environment. Data exchange with FMUs for CS is limited to discrete communication points, in the time between two communication points the encapsulated system model is solved by the FMU's internal solver.
- FMI for Model Exchange (ME) provides access to a numeric representation of a model's underlying mathematical equations, unlike the black-box approach pursued with FMI for CS. It supports models that are described by discrete, algebraic and differential equations and allows to define time-events, state-events and step-events. In case an FMU for ME represents a time-continuous system, the embedding simulation environment is itself responsible for solving the model equations.

An FMU consists of a ZIP file containing an XML-based model description together with a shared library and/or source code implementing the corresponding interface (as C API). In the case of FMUs for CS the shared library may be either a self-contained simulation component or call another simulation tool at run-time (tool coupling).

The number of simulation tools that claim to support FMI is ever growing, but they do so in a variety of different ways. For instance, while some only offer to import FMUs, others are also able to export their models and functionality. Furthermore, the way a simulation tool utilizes the functionality provided by FMUs is strongly related to the simulation paradigm that the tool is based on. For instance, a tool designed for the co-simulation of (physical) models with continuous outputs (cp. [4]) typically focuses on a different subset of FMI's functionality than a tool designed for the deterministic simulation of heterogeneous systems (cp. [5]).

III. FMI-based Simulation of Closed-loop Control Systems

Fig. 1 shows the typical graphical representation of a closed-loop control system, consisting of a physical plant and a controller. For engineering applications such types of models are of obvious importance, as they provide a generic and widely applicable control mechanism. Hence, it is important to understand how to simulate such models properly, also within the context of FMI-based co-simulation and model exchange approaches.

However, co-simulation models containing loops always require special attention. While for instance loops representing the (acausal) coupling of physical models can be dealt with the help of advanced numerical methods [6], it is in general not possible to resolve them in a meaningful way [7]. Therefore, a detailed and unambiguous description of the proper execution and data flow of FMI-based hybrid closed-loop control models is presented in the following.

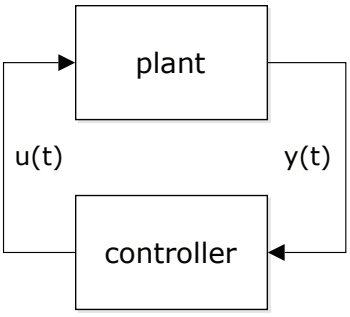

Fig. 1. Closed-loop control model example.

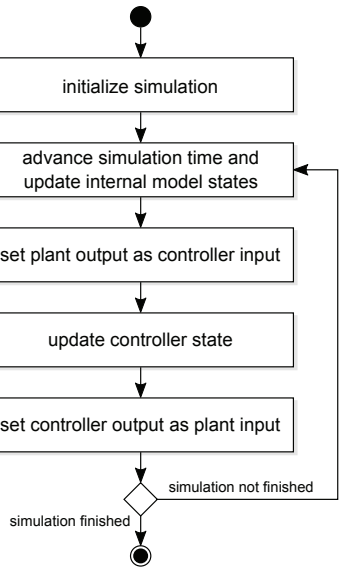

Fig. 2. Schematics of simulation execution and data flow for hybrid FMI-based closed-loop control models.

Fig. 2 shows the schematics of the simulation execution and data flow of a closed-loop control system. It is assumed that either

- the plant model and the controller model are represented by individual FMUs or that
- one of the two models is represented by an FMU, whereas the other model is directly implemented in the hosting simulation environment.

Furthermore, at least one FMU is expected to be an FMU for CS, otherwise the overall model could not be considered a co-simulation model. Finally, it is assumed that the output of the plant model is strictly continuous with respect to time (variability *continuous*), while the controller output is piecewise constant (variability *discrete*).

Even though the schematics in Fig. 2 seem intuitive, there are several FMI-specific details to consider:

- *Advancing simulation time* and *updating model states* can in general not be decoupled. Unless very simple co-simulation approaches are applied—like for instance fixed step-size co-simulation—various factors depending on the internal model states may have an impact on the

step-size control, especially state error estimates [6] and state-events [8].

- The plant and the controller exchange data at *discrete synchronization points*. Since the output of the plant model is continuous with respect to time, an altered controller output at a synchronization point will not immediately change the plant's output at that given point in time. Rather the effect of the altered controller output will have an impact on the plant's model state during the next time advance. Hence, from a data-flow perspective, the model's loop can be reduced to the sequence described in Fig. 2, resolving any potential infinite regression.
- At each synchronization point, the *internal controller state is updated* upon receiving the output from the plant.[1] Since this update needs to happen instantaneously (i.e., without advancing simulation time) it has to be treated as an event for FMUs for ME, whereas FMUs for CS need to iterate on the new input. Unfortunately, in contrast to FMI 1.0 where a communication step size of zero was specifically intended for event iteration, the current FMI 2.0 version explicitly requires the communication step size to be greater than zero. This effectively renders FMUs for CS in the current version 2.0 unfeasible for closed-loop controller models.

Without giving rigorous proof here, it needs to be emphasized that even seemingly small deviations from the execution and data flow procedure described above can cause distinctively erroneous simulation results (cp. results in Section V).

IV. Reference Model Definition

This section introduces a simple model that will be used for a comparative study on FMI-based co-simulation of hybrid closed-loop control systems. In this specific case, the physical plant is a thermal room model with a controller in a feedback-loop, which reacts on the room's air temperature in order to turn on/off a heater.

The model comes from the need to unify the domains of control engineering and building simulation. Both usually have their own tools, and moving the whole domain knowledge into one of the respective tools would be too time consuming and prone to modeling errors. Co-simulation therefore is a viable approach, but the quality of the results has to be ensured, as the tools used in the different domains have different requirements.

The plant model is a first order ODE, representing the thermal model of a room:

$$\dot{T}_{\text{room}} = \begin{cases} -Q_{\text{loss}} & \text{if heater is off,} \\ Q_{\text{heater}} - Q_{\text{loss}} & \text{if heater is on.} \end{cases} \quad (1)$$

T_{room} is the room air temperature, Q_{loss} the difference between losses to the environment and inner loads, and Q_{heater} is the power of the heating unit. Both Q_{loss} and Q_{heater} are normalized by the thermal capacity of the room air.

[1]Without loss of generality, as delays can be accounted for within the controller model itself and its impact on the synchronization point schedule.

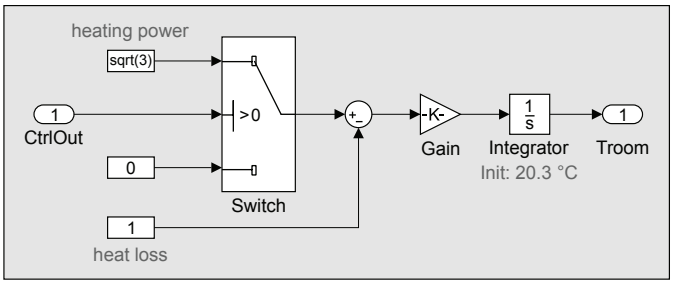

Fig. 3. Room model and heating unit implemented in Simulink.

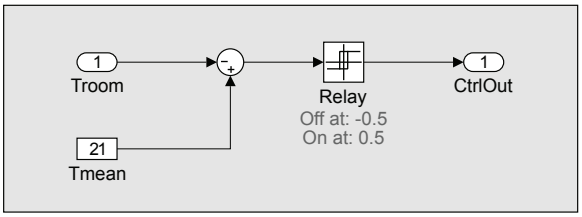

Fig. 4. Thermostat controller model implemented in Simulink.

The heater is controlled by a thermostat set to 21°C with a dead-band of 1°C. It switches on if the room air temperature T_{room} is below 20.5°C and off if 21.5°C is exceeded.

While simple, the overall model comprises two important features inherent to many closed-loop control systems:

1) The plant is described by a continuous time-driven model (ODE), whereas the controller is represented by a discrete event-driven model (threshold crossings).
2) The plant's representation effectively switches between two distinct models, depending on the possible states of the controller output (on/off).

Even though real-world applications would rely upon considerably more elaborate domain models, their respective complexity would be hidden behind the respective interfaces. As such, this simple model provides a reasonable and meaningful reference, featuring the distinct characteristics of hybrid closed-loop control models.

V. Comparative Analysis of Simulation Approaches

In the following, the model from the previous section is applied to actual simulation environments. In this particular example, Simulink[2] is applied for control design and TRNSYS[3] for building and HVAC (heating, ventilation and air conditioning) simulation, respectively. These two were chosen as they are both industrial-grade simulation tools in their respective domains. While Simulink is designed for precise simulations on arbitrary timescales, TRNSYS is primarily intended for yearly simulations of complex buildings an their energy systems at a fixed time step of usually 15 minutes or 1 hour.

[2]MATLAB/Simulink 8.1, see `http://www.mathworks.com/`.
[3]TRNSYS 17, see `http://www.trnsys.com/`.

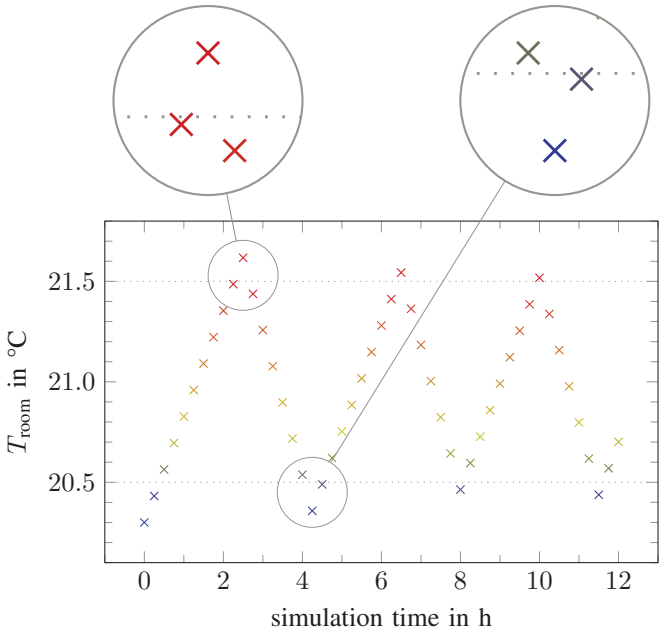

Fig. 5. Simulation results from reference implementation.

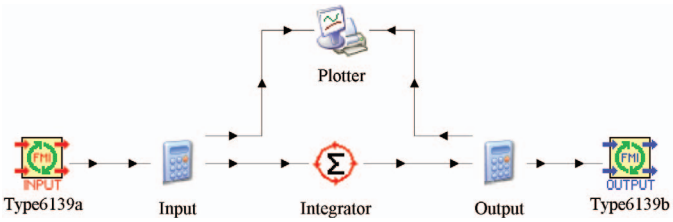

Fig. 6. Room model and heating unit implemented in TRNSYS.

A. Reference implementation

For reference, the model was implemented in Simulink only. The full system was modeled according to Fig. 1, with both blocks directly implemented as Simulink sub-models. The ODE from Eq. 1 was modeled using a *Switch* block, in order to account for the two states associated to the possible controller outputs, together with a *Gain* block and an *Integrator* block (see Fig. 3). An integrator with a fixed step-size of 900 s was chosen to emulate a behavior consistent with a fixed-step co-simulation approach. The ratio of Q_{loss} to Q_{heater} was chosen as $-1:\sqrt{3}$ to avoid coincidental temperature threshold crossings at exact sampling times. The thermostat controlling the room air temperature was modeled using a *Relay* block, providing a simple hysteresis model (see Fig. 4).

Fig. 5 shows simulation results for this reference implementation. An overshooting of the room air temperature T_{room} with respect to the controller's thresholds can be clearly observed. Only once the temperature has crossed a controller threshold after a full integration step, the heater is turned on/off. This behavior is expected as a direct result of the utilized fixed step-size solver, which executes the controller loop only at the end of each integration step.

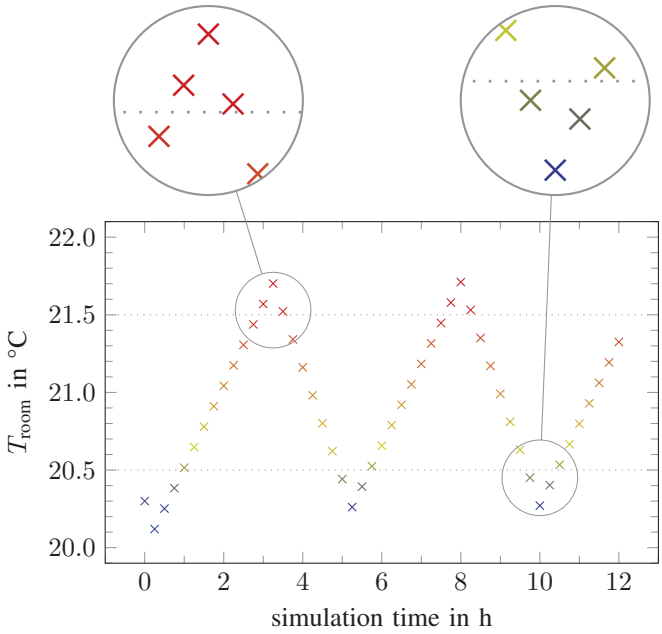

Fig. 7. Simulation results from Simulink-based co-simulation.

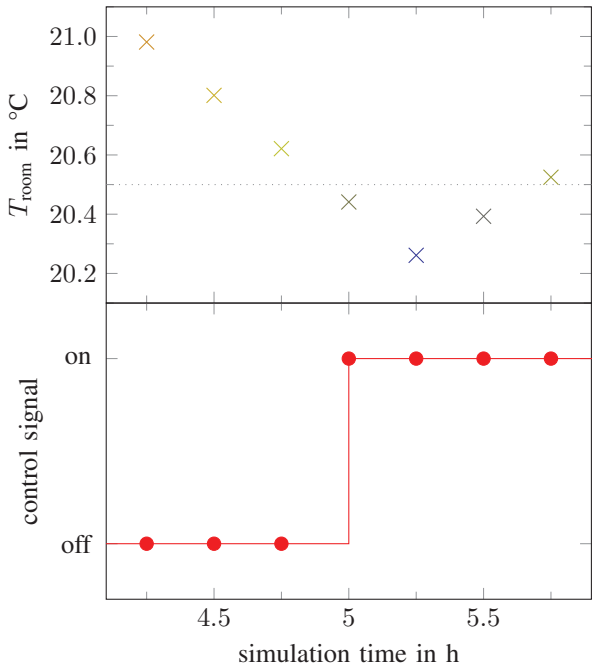

Fig. 8. Simulation results detail from Simulink-based co-simulation.

B. Simulink-based co-simulation approach

This example studies a straightforward coupling approach, where Simulink is used as main application, importing a TRNSYS model as an FMU for CS. Again, the full system was modeled according to Fig. 1, with the block associated to the controller directly implemented as Simulink sub-model. However, the block representing the plant model imported a TRNSYS model as FMU for CS with the help of the Modelon

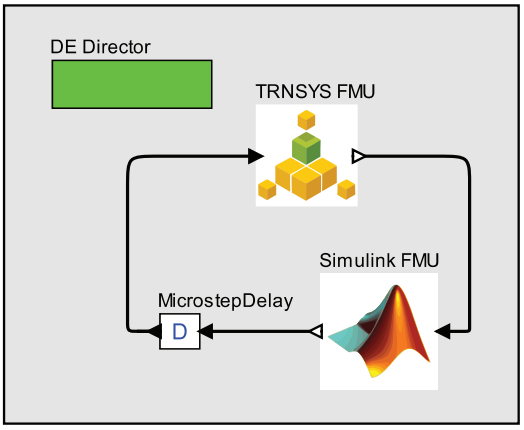

Fig. 9. FUMOLA co-simulation model.

FMI Toolbox[4].

Even though TRNSYS itself does not officially provide FMI support, there is a dedicated FMI-based tool coupling solution openly available[5] that has been deployed here. It provides dedicated TRNSYS blocks, referred to as *Type6139*, that allow to establish an FMI-based coupling with TRNSYS at run-time with a fixed, user-defined communication step-size. Apart from the additional input and output block of this type, TRNSYS models can be constructed in the usual way (see Fig. 6). In this case, the response to the control signal was implemented using a calculator block. It simply replaces the switch block from the Simulink reference model with a multiplication of the term Q_{heater} by either 0 or 1, corresponding to controller state off or on.

Fig. 7 shows the result from a simulation run with a fixed communication step size of 900 s. It is clearly visible that the overshooting of the room air temperature T_{room} is more pronounced than for the case of the reference implementation, obviously failing to reproduce the same result (cp. Fig. 5). A closer look at Fig. 8, which shows a detailed view of not only T_{room} but also the control signal, reveals that the response of the plant to the controller's signal is delayed by exactly one simulation step.

Since both Simulink and the Modelon FMI Toolbox are proprietary and closed-source, there are no details available regarding their implementation. Thus, any conclusion concerning possible reasons for this behavior are purely speculative. However, it is clear that in some way the execution and data flow scheme presented in Section III is not met. Most probably, either the sequence of the data flow is reversed, or the input to the imported TRNSYS FMU is not set before but only after its time advance.

C. FUMOLA-based co-simulation approach

This example showcases the implementation of above reference model in a dedicated co-simulation framework, importing

[4]Modelon FMI Toolbox 2.1, see `http://www.modelon.com/`.

[5]FMI++ export utility, see `http://fmipp.sourceforge.net/`.

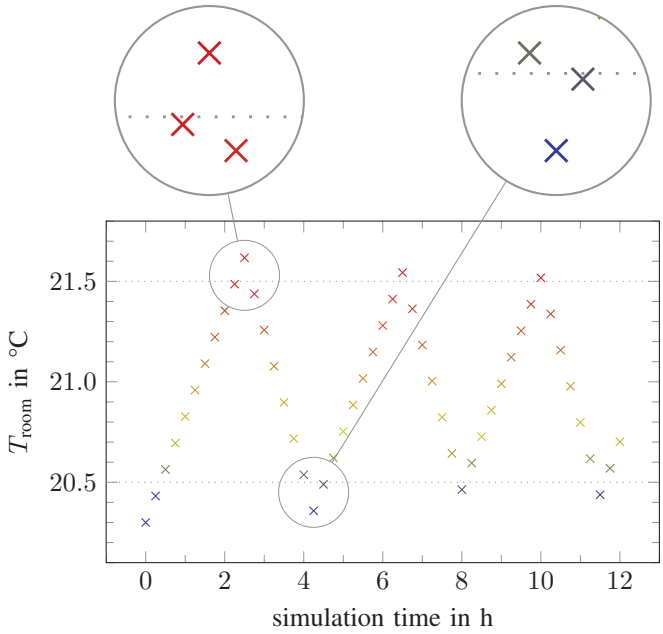

Fig. 10. Simulation results from FUMOLA-based co-simulation.

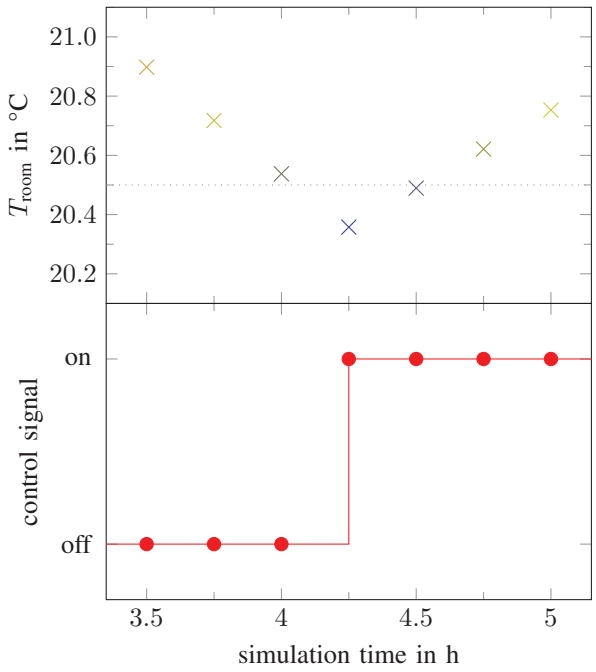

Fig. 11. Simulation results detail from FUMOLA-based co-simulation.

both the plant as well as the controller as FMUs. To this end FUMOLA[6] is used, a co-simulation environment specifically designed to support the features offered by the FMI specification [9], developed on top of the Ptolemy II simulation environment [10]. Like above, the plant was again modeled in TRNSYS and imported as FMU for CS. The controller was modeled in Simulink (see Fig.4) and then exported as FMU

[6]FUMOLA, see `http://fumola.sourceforge.net/`

for ME with the help of a dedicated Simulink Coder Target developed by Dassault Systèmes for the export of FMUs from Simulink[7].

In principle, the full system was again modeled according to Fig. 1. However, the graphical representation of the FUMOLA co-simulation model reveals some additional features (see Fig. 9). The first additional feature is the *DE Director* block, which indicates that the model is executed as an discrete event system [11]. In the case of FUMOLA, this means that the blocks representing FMUs are (from a conceptual point of view) executed as concurrent processes that signal requests for synchronization as time-discrete events. The second additional feature is the *MicrostepDelay* block, which here effectively resolves the model's loop according to the data flow sequence defined in Fig. 2.

Fig. 10 shows the results of a FUMOLA-based co-simulation run. When compared to Fig. 5 it becomes obvious that these results are identical to the results from the Simulink-only reference implementation. The exact analysis reveals that the results are indeed identical in the first 10 digits (apart from rounding errors). The simulation results detail shown in Fig. 11 illustrates that the response of the plant model in fact follows the control output as expected. Even though not a rigorous proof, this strongly indicates that the applied discrete event-based modeling paradigm fully captures the requirements given in Section III.

VI. Conclusions and Outlook

The successful coupling of simulation software from different domains is a step forward for the easier transfer of knowledge between those domains. While this gives experts the possibility to rely on the tools they are already comfortable with, they still need to specify clear interfaces to the other domains. The Functional Mock-up Interface specification is currently the de facto standard that provides the technical background for such interfaces.

This article has illustrated the applicability of the FMI specification for the co-simulation of hybrid closed-loop control system models, where a continuous time-based plant model is connected to a discrete event-driven controller model. The semantics of execution and data-flow of such models have been discussed, with a special focus on FMI-related features. The results demonstrate that FMI-based co-simulation approaches are very well capable of providing the functionality needed for such models. However, not even popular state-of-the-art simulation environments are necessarily able to do so properly.

This example points out the need to further investigate the possibilities offered by the FMI specification. Even though left out from the FMI specification on purpose, the systematic treatment of the execution of FMI-based co-simulation models needs a more formal framework. This would give the opportunity to define references for the validation of simulation environments, in order to avoid improper implementation and usage.

Acknowledgments

Part of this work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 will develop and demonstrate new generation computational tools for building and community energy systems based on Modelica, Functional Mock-up Interface and BIM standards.

References

[1] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 44, no. 3, pp. 318–326, 2014.

[2] T. Blochwitz, M. Otter *et al.*, "The Functional Mockup Interface for Tool independent Exchange of Simulation Models," in *Proceedings of the 8th International Modelica Conference*, 2011.

[3] A. Elsheikh, M. Awais *et al.*, "Modelica-enabled rapid prototyping of cyber-physical energy systems via the Functional Mockup Interface," in *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, May 2013, pp. 1–6.

[4] V. Galtier, S. Vialle *et al.*, "FMI-Based Distributed Multi-Simulation with DACCOSIM," in *Symposium on Theory of Modeling and Simulation (TMS'15)*, 2015, pp. 804–811.

[5] S. Tripakis and D. Broman, "Bridging the Semantic Gap Between Heterogeneous Modeling Formalisms and FMI," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-30, April 2014.

[6] S. Sicklinger, V. Belsky *et al.*, "Interface jacobian-based co-simulation," *International Journal for Numerical Methods in Engineering*, vol. 98, no. 6, pp. 418–444, 2014.

[7] D. Broman, C. Brooks *et al.*, "Determinate Composition of FMUs for Co-simulation," in *Proceedings of the Eleventh ACM International Conference on Embedded Software (EMSOFT '13)*, 2013, pp. 2:1–2:12.

[8] W. Müller and E. Widl, "Using FMI components in discrete event systems," in *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2015 Workshop on*, April 2015, pp. 1–6.

[9] E. Widl, W. Müller *et al.*, "Simulation of multi-domain energy systems based on the functional mock-up interface specification," in *Smart Electric Distribution Systems and Technologies (EDST), 2015 International Symposium on*, October 2015.

[10] J. Eker, J. Janneck *et al.*, "Taming heterogeneity - the Ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127 – 144, 2003.

[11] E. A. Lee, "Modeling concurrent real-time processes using discrete events." *Ann. Software Eng.*, vol. 7, pp. 25–45, 1999.

[7]FMU Export from Simulink 2.1, see `http://www.3ds.com/`.