A co-simulation framework interoperability for Neo-campus project

Yassine Motie, Alexandre Nketsa, Philippe Truillet

A co-simulation framework interoperability for Neo-campus project

Yassine Motie, Alexandre Nketsa, Philippe Truillet
LAAS-CNRS, University of Toulouse, CNRS, IRIT, UPS, Toulouse, France
email: Yassine.Motie@irit.fr, alex@laas.fr, philippe.truillet@irit.fr

KEYWORDS
Complex systems; interoperability; Mediation; co-simulation; FMI; Neo-campus

ABSTRACT

It is common accepted that complex systems or cyber-physical systems need co-simulation for their study. Further more, they are made of heterogeneous sub-systems that have to exchange data. Usually each sub-system is modeled using specific tools, environments and simulators. The simulators have to interoperate to realize all the simulation of the system. It is known that interoperativity is a broad and complex subject. Interoperability is a strong commitment as the communication solution in heterogeneous systems. This paper describes a co-simulation framework interoperability based FMI (Functional Mock up Interface) standard for the structural part and data mediation for semantic part. We present a case study for Neo-Campus project that shows how the framework helps to build the semantic interoperability of a cyberphysical system.

INTRODUCTION

The Neo-Campus project (Gleizes et al. 2017), supported by the University of Toulouse III, aims to link the skills of researchers from different fields of the University to design the campus of the future. Three major areas are identified : facilitating the life of the campus user, reducing the ecological impact and controlling energy consumption. The campus is considered as a smart city where several thousand data streams come from heterogeneous sensors placed inside and outside the buildings (CO2, wind, humidity, luminosity, human presence, energy and fluid consumption , ...). We distinguish :

- Raw data : These are the energy consumption data (water, electricity, gas).
- Activity-specific data : These are post-processed data resulting from the merging of raw data (pedagogical activities, room occupancy, ..).
- Incident-specific data : These are the failures identified on campus (heating of computer equipment, network failures, ...)
- The ambient data : This concerns the context in which the scenario takes place (temperature, weather, CO2 level in the air).

We have built a knowledge base from sensors data that provides real-time data and relationship between them in order to be used for simulation for example.

Each expert working in Neo-Campus project interact with data differently using a specific field within simulation, hence the need to build a co-simulation framework in order to ease the collaboration.

The paper is organized as follows, first we present the related works on interoperability and our approach to build the co-simulation framework interoperability. We continue with the Neo-Campus use case and give a conclusion and future work.

RELATED WORKS

Interoperability can be defined as the ability of two or more entities to communicate and cooperate despite differences in the implementation language, the execution environment, or the model abstraction (Kohar et al. 1996). Interoperability is a complex problem. There are many ways to deal with it. We have identified two main approaches : (1) based on levels of conceptual interoperability models (LCIM), (2) based on structural and semantic interoperabilities. In (Diallo et al. 2011), LCIM is used as the theoretical backbone for developing and implementing an interoperability framework that supports the exchange of XML-based languages. They defined 7 levels of LCIM with the goal to separate model, simulation, and simulator in order to better understand how to make models interoperate. The authors of (Rezaei et al. 2013) presented an overview of the development of interoperability assessment models. They proposed an approach to measure the interoperability and used four interoperability levels to define a metric. (Li et al. 2013) used comparisons between reusability and interoperability, composability and interoperability to show the importance of interoperability. The authors proposed to use models to build interoperability. Thus they decomposed interoperability into: (1) technical (or structural) interoperability (communication ports) (2) and substantive (or semantic) interoperability (contents meaning) It means that simulation sub-systems can talk to each other and exchange data, but to understand

each other correctly and co-simulate effectively requires substantive interoperability. We agree with this decomposition but instead of using models, which imposes an important customization of the exported model, we take advantage of a known simulation standard, FMI (Functional Mock up Interface) to build the structural part. Our semantic interoperability will be based on data mediation.

CO-SIMULATION FRAMEWORK INTEROPERABILITY APPROACH

Our approach for the design of a co-simulation framework interoperability is based on :

1. a **co-simulation**
2. a **software components** approach which is defined by (Szyperski 1996) as a unit of composition with contractually specified interfaces and explicit context dependencies only
3. a **Structural interoperability** using the standardized interface FMI (Functional Mock-up interface)
4. a **semantical interoperability** using mediation for adaptation of the data

Co-simulation

Co-simulation is defined as the coupling of several simulation tools (Hessel et al. 1999) where each tool handles part of a modular problem where data exchange is restricted to discrete communication points and where subsystems are resolved independently between these points. This allows each designer to interact with the complex system in order to retain its business expertise and continue to use its own digital tools.

From the literature, we have constructed a global scheme of co-simulation in order to have an overall view of it (cf. Figure 1). The models are described by their interfaces without any access to their contents. Aiming at securing model exchanging between designers and ensuring privacy and robustness, we went for a black box model interoperability. This last makes it then possible to exchange and use the information between those components.

Component approach

We chose a component approach in order to overcome the limits of the white box approach which consists on redeveloping of all the heterogeneous subsystems in the same language (Kossel et al. 2006). This imposes that

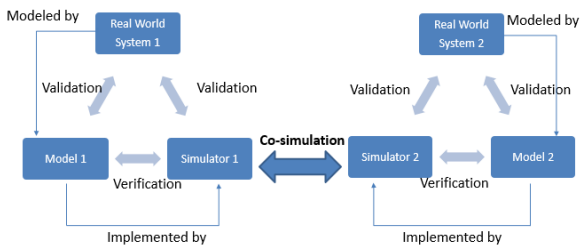

Figure 1: Overview of co-simulation of two subsystems

models developped by different designers are transparent and accessible (Allain et al. 2002). The component approach makes it possible to co-operate prefabricated pieces, perhaps developed at different times, by different people, and possibly with different uses in mind. The main reason is to improve the flexibility, reliability, and reusability of our framework due to the (re)use of software components already tested and validated avoiding risks of robustness. A component is an autonomous deployment entity which encapsulates the software code showing only its interfaces. An interface can be described as a service abstraction, that defines the operations that the service supports, independently from any particular implementation (Lea and Marlowe 1995). Each component should provide the way how it can be generated (plug-out) either from a white box model, or another black box provided by a simulator. (cf. Figure 2) represents a communication between two software components; componentA (as a piece of software) and componentB (UML notation). This component

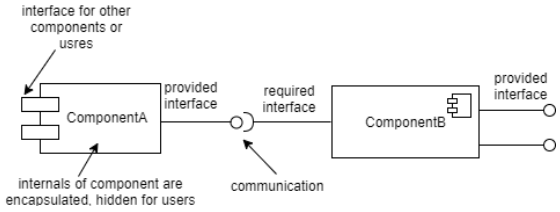

Figure 2: interaction between two Components

approach should allow data mapping between models, applications and several building simulators (Simulink, Dymola, Saber...) using it. A standard interface would be used to warranty the compatibility between these tools.

Structural interoperability based FMI

Structural interoperability requests to define communication ports and has to guarantee the possibility of connection by respecting data types and the direction of the ports.

Structural Interoperability Standard - FMI

We chose the standard FMI (Blochwitz et al. 2011) see (cf. Figure 3) which uses a master-slave architecture as a simulation interoperability standard.

FMI : Functional Mockup Interface is a Standard interface for the solution of coupled time dependent systems, consisting of continuous or discrete time subsystems. It provides interfaces between master and slaves and addresses both data exchange and algorithmic problems. Simple and sophisticated master algorithms are supported. However, the master algorithm itself is not part of FMI for Co-Simulation and should be defined (Consortium et al. 2010) (Enge-Rosenblatt et al. 2011). FMI supports different working modes, in particularly:

1. FMI for model exchange (when modeling environment can generate C code of a dynamic system model that can be utilized by other modeling and simulation environments)

2. FMI for co-simulation (when an interface standard is provided for coupling of simulation tools in a co-simulation environment)

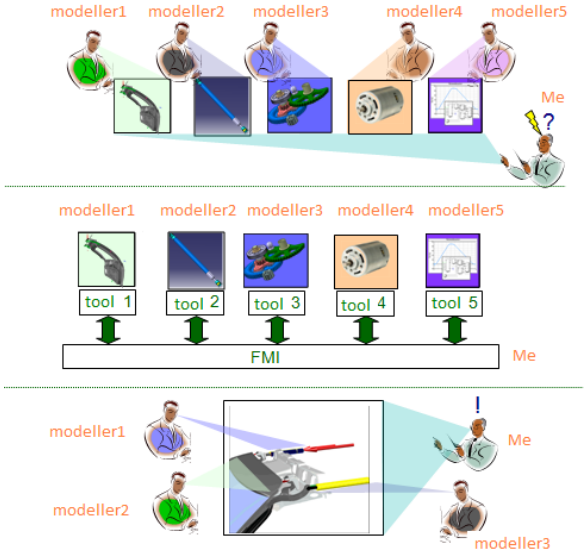

Figure 3: Interoperability using FMI

The use of FMI can be summarized in 4 steps:

- **The design step** :The package of the model of simulation in one component FMU which summarizes in the modeling (creation of the model of simulation), And transformation (publication of the FMU which contains an xml file and the model code or its file), dealing with the main challenge, which is the gap between the semantics of the source formalism of the various calculation models (state machines, Discrete event, data flow or timed automata) and the semantics of FMI (Tripakis 2015).

- **The composition step** : The model of the subsystem is joined to the complex system by establishing the connection graph of the simulation components

- **The deployment step** : The FMUs are made available to the slave simulators. This can be made offline (manually by the user) or online (automatically by the master and where the user specifies in which network the instances of the FMUs are transferred)

- **The simulation step** : The master is responsible for the life cycle of FMUs instances during the execution of the simulation

This choice is motivated by the fact that FMI is a standard and therefore minimizes the customization of the exported model. It is a tool independent that facilitates the exchange of models between different tools and which therefore minimizes the effort of integration by proposing approaches that are specific to it.

Semantical interoperability based data mediation

The interoperability is the ability to share information between systems and applications in meaningful ways. While most system engineering or system applications stop at the structural level, assuming that if you can read it you're going to understand it. Additional level of interoperability and the next that really matters for us is a semantic one which needs common information model to be defined for exchanging the meaning of information. Then the content of the information exchanged is unambiguously defined.

This approach uses a model that describes the information shared by taking their semantics into account in the form of contexts of use. The mediation model leads to define three types of integration rules:

1. constraint rules that reduce the objects to be considered according to predicates,

2. merge rules that aggregate instances of classes of similar,

3. and join rules that combine information from multiple object classes based on one or more common properties.

We distinguish two types of mediation:

1. Schema mediation which provides better extensibility and often better scalability (object interfaces, rule-based language).

2. Context mediation seeks to discover data that is semantically close, it is able to locate and adapt information to ensure complete transparency. We can therefore take advantage of the robustness of the

mediation schema approach and combine it with the semantic approximation techniques of context mediation. This semantic mediation will be used to correlate, aggregate and dispatch data with respect to the control that we want to enforce on data produced or consumed.

For example, one component may produce data with meaning T0, while another may consume data with T1. It may be that there is no direct T0T1 bridge, but there are separated T0T' and T'T1 bridges. A mediator is required to assemble the bridges to complete the T0T1 translation. Our solution is based on formal interface descriptions. When a simple component has an input or output event as a kind of interaction, our interface description will list those events including their (typed) parameters. It could be done automatically if a generator tool, based on language mapping, processes an interface description and produces a proxy (environment side stub) and a driver (component-side stub) for the component. Proxy and driver communicate through a mediation channel, using a protocol for message exchange, see (cf. Figure 4). At the present version, mediators are hand-coded.

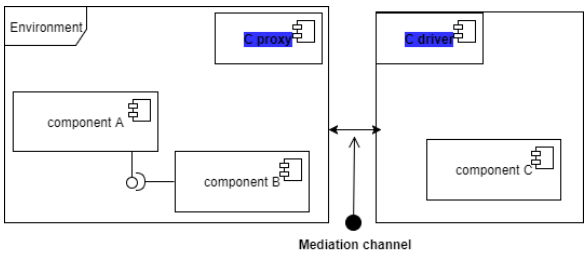

Figure 4: components mediation

The co-simulation framework based component

Simulator

We chose the CosiMate environment which offers a complete simulation environment dynamically linking heterogeneous simulators and which can be extended to different simulation environments on different platforms. CosiMate provides synchronization methods that take into account the different behaviors of the languages and the simulators used. Thus, when a simulation is performed on a network, CosiMate considers the intrinsic constraints of the communication medium. CosiMate adapts to the network configuration, offering a co-simulation based on a multi-client multi-server to avoid unnecessary communications between simulators instantiating local routers for each computer in the co-simulation. And when different parts of the system are co-simulated at different levels of abstraction, it is necessary to add adapters (wrappers) to the compatibility of data exchange between models at different levels. CosiMate meets our needs by offering two co-simulation working modes:

- **Event mode :** The router (that manages the data exchange and synchronizes the simulator) does not deal with any notion of time. This mode of communication makes it possible to establish a connection between the event simulators (HDL simulators, UML models) and sequential simulators (code C for example). The data is transmitted once available on the CosiMate bus. CosiMate is flexible enough to support different communication protocols.The data is transmitted once available on the CosiMate bus, the valid transmission of the router between the sender and receiver (does not check if the recipient has read the data). CosiMate is flexible enough to support different communication protocols.
- **Synchronized mode :** : The router synchronizes the models taking the minimum time. This mode is suitable for simulation engines using solvers (such as Matlab / Simulink).

Cosimate-FMI

Cosimate allows the co-simulation between FMI and also non-FMI models. There are simulators which are not supported by Cosimate, thus the need to wrap them as an FMU component in order to plug them to our cosimate bus

System

The co-simulation of a complex system can thus be based on the joint simulation of all its subsystems. It also makes it possible to simulate the whole system by coordinating and exchanging data calculated and interpreted by each subsystem, in order to obtain a result which does not modify the functionality of the implementation of the future system. Among our different simulated models, we distinguish:

1. Functional simulation allowing the validation of the aspects of the system which are independent of time, and here we can dissociate the sequential simulators and the event simulators.
2. Temporal simulation which aims to exchange data in time windows. Knowing that if data is not consumed in a given time window, it may be lost. A synchronization model is therefore necessary to coordinate the parallel execution of our different simulators.

We mention the master-slave model (cf. Figure 5), comprising a master simulation and one or more slave simulations. In this case, the slave simulators are executed using procedure calls, which results in an inability to execute them simultaneously. The distributed model overcomes the limitations of the master-slave model, which

relies on a co-simulation bus used as a communication protocol (cf. Figure 6). The complexity of this model focuses on the co-simulation platform: managing access to the co-simulation bus and coordinating the data by the bus controller. Another great difficulty comes from the integration of time (Yoo and Choi 1997) which is different between embedded software systems, hardware and the surrounding environment.

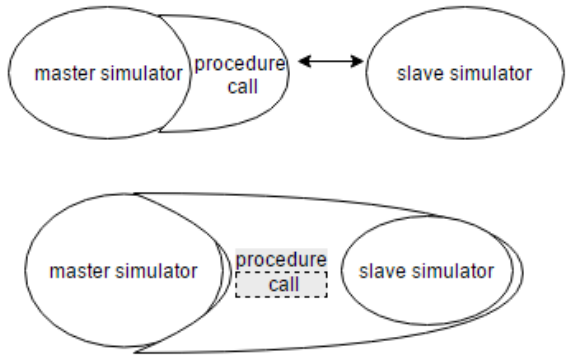

Figure 5: Example of a master-slave co-simulation platform

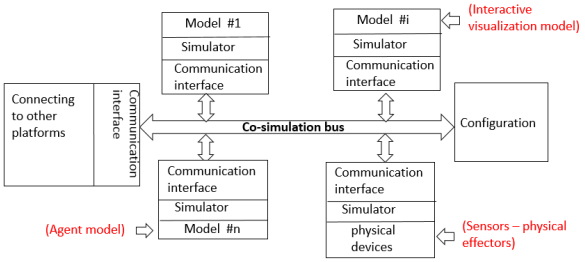

Figure 6: Example of a distributed co-simulation platform

NEO-CAMPUS CASE STUDY

System description

As described in the introduction there are several simulators and sensors scattered around the campus (cf. Figure 7)

Different simulators used

We therefore have: In one hand several simulators on a different fields using different kind of data. One is working with **Matlab Simulink** on the energetic consumption using a black box neural network Heat pump model to ensure a comfortable desired in the rooms by heating and cooling when it is necessary. It is interacting with the outdoors getting from sensors the Electric power (Kw) and the temperature coming from the building and generating with a specific time step. The second simulator is working with **powersims** toolbox of Simulink (Khader et al. 2011) using Maximum Power Point Tracking making it possible to follow the maximum power point of a non-linear electric generator. It is interacting with the outdoors getting the values of the Photovoltaic current and the voltage and generates Converter control setpoint. The third simulator using **Contiki** (Dunkels et al. 2004) which is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of things devices using Cooja which allows large and small networks of Contiki motes to be simulated in order to evaluate the performances (energy, delays) of IOT networks using the protocol CCN (content centric Networking) applied on a network of sensors. Developed in C++ under linux OS. The way it interacts with the outdoors is using interest (requests sent by users containing the name of the data such as the temperature) and generating the value of this data. In the other hand the **collection of sensors data** is stored in a NoSQL **database** (mongodb).

Co-simulation engine

The design approach of Neo-campus is necessarily scalable and adaptive, which directs our work towards the development of global and open simulation environment. As we said before we adopted the component approach and described the general FMI's way of working. This last follows a master slave architecture, and we mentioned that a master algorithm needs to be defined in order to synchronize the simulation of all subsystems and to proceed in communication steps, that the data exchange between subsystems is connected via MPI, TCP/IP, Sockets, and that the mapping between outputs to inputs has to be initialized. Cosimate, as described, makes us save the efforts of dealing with synchronization between our subsystems. To perform this integration, CosiMate provides libraries to make the cus-

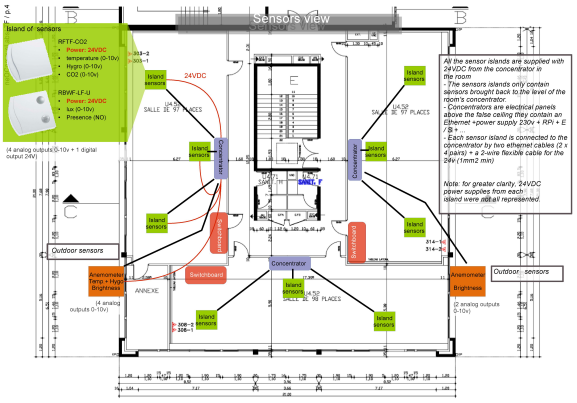

Figure 7: View of neo-campus sensors

tomization easier. The libraries contain (1) I/O ports compiled and described for the simulator/language used. (2) I/O ports description. This description depends on the environment in which the ports are to be used: for example, a header file for C/C++ language is provided. In our case and according to the different simulators mentioned previously, we constructed an FMU's component for each one either by using FMI toolbox like for MATLAB/Simulink or PSIM, or a wrapper using FMI Library from Modelon for the simulator using Contiki. We made it easy to connect all the simulators knowing that for example, Simulink and PSIM are supported by Cosimate but Contiki is not. But the CosiMate FMI connector can load and run all FMI models compatible with FMI 1.0/2.0 for the Co-simulation mode. As we said before each of our FMU files is a zip file that contains a file named modelDescription.xml and one or more platform-dependent shared libraries. The XML files are used to describe how a model running in a simulation environment is connected to the CosiMate bus. We should mention that CosiMate allows execution in the native simulation environment, users can easily work in their familiar environment controlling, debugging, and monitoring simulations as if they are running in a stand alone mode integration. We can also use remote procedure: if the model is to be run on a remote machine. The CosiMate Spy tool is used to monitor and control the co-simulation components and processes. It acts as a reader of the CosiMate bus without modifying data exchanges or simulations synchronization during the co-simulation.

Mediator components

One of the problems encountered is the mediation part, since we want to achieve a semantic interoperability we offered the possibility for each simulator to decide of the way it wants to receive information and depending of the components it's talking to (if it already knows them) to convert its output. For that we encapsulate a mediator with each component before connecting it to the cosimate bus. We added some procedures which allow us to copy and later restore the complete state of an FMU component providing a mechanism for rollback (inspired from the optional functions of the API of FMI 2.0). For our sensor network and as we said that it uses a database to store raw data. This has led us to develop a java simulator (using Mongodb Java Driver) that bridges between the mangodb database and our cosimate bus. We encapsulate the database and our simulator using JFMI (a java wrapper for FMI). So this virtual encapsulation offers capabilities of data mediation and distributes query processing. So the other simulators have no need to know about the database type and location and data can be accessed easily see (cf. Figure 8).

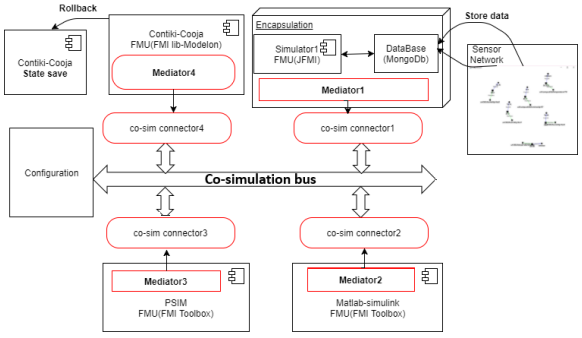

Figure 8: components mediation

CONCLUSION

We have implemented our architecture and our modeling works well, we took as example the 4 simulators including Contiki which is not compatible with Cosimate and for which we had to generate a slave FMU. Our database was encapsulated using JFMI. We were able to solve not only these structural problems but we added mediators to our platform in order to achieve semantic interoperability. It is necessary to mention that our framework allows the integration of all types of simulators and that for the non FMI and even if they are not supported by cosimate the use of a wrapper is enough to envelop them with a c code in order to connect them to the cosimate bus.
This work allowed us to first make an inventory of the practices of the various actors of the neOCampus project. In order to allow the various experts to communicate and collaborate, we realized that it was preferable for them to keep their own practices by allowing them to build or improve their own "expert" simulator. Thus, the objective is a completely open system, easy to use, accepting all types of simulators.
As future work, we would like to build a tool for the generation of mediator. Moreover, we would like to approach semantic interoperability using ontology for the comparison purposes.

REFERENCES

Allain S.; Chateau J.P.; and Bouaziz O., 2002. *Constitutive model of the TWIP effect in a polycrystalline high manganese content austenitic steel. steel research international*, 73, no. 6-7, 299–302.

Blochwitz T.; Otter M.; Arnold M.; Bausch C.; Elmqvist H.; Junghanns A.; Mauß J.; Monteiro M.; Neidhold T.; Neumerkel D.; et al., 2011. *The functional mockup interface for tool independent exchange of simulation models.* In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany.* Linköping University Electronic Press, 063, 105–114.

Consortium M. et al., 2010. *Functional Mock-up Interface for Co-Simulation. Accessed March*, 1, 2013.

Diallo S.Y.; Tolk A.; Graff J.; and Barraco A., 2011. *Using the levels of conceptual interoperability model and model-based data engineering to develop a modular interoperability framework.* In *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2576–2586.

Dunkels A.; Gronvall B.; and Voigt T., 2004. *Contiki-a lightweight and flexible operating system for tiny networked sensors.* In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 455–462.

Enge-Rosenblatt O.; Clauß C.; Schneider A.; and Schneider P., 2011. *Functional Digital Mock-up and the Functional Mock-up Interface-Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems.* In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*. Linköping University Electronic Press, 063, 748–755.

Gleizes M.P.; Boes J.; Lartigue B.; and Thiébolt F., 2017. *neOCampus: A Demonstrator of Connected, Innovative, Intelligent and Sustainable Campus.* In *International Conference on Intelligent Interactive Multimedia Systems and Services*. Springer, 482–491.

Hessel F.; Le Marrec P.; Valderrama C.A.; Romdhani M.; and Jerraya A.A., 1999. *MCImultilanguage distributed co-simulation tool.* In *Distributed and Parallel Embedded Systems*, Springer. 191–200.

Khader S.; Hadad A.; and Abu-Aisheh A.A., 2011. *The Application of PSIM & MATLAB/SIMULINK in power electronics courses.* In *Global Engineering Education Conference (EDUCON), 2011 IEEE*. IEEE, 118–121.

Kohar H.; Wegner P.; and Smit L., 1996. *System for setting ambient parameters.* US Patent 5,554,979.

Kossel R.; Tegethoff W.; Bodmann M.; and Lemke N., 2006. *Simulation of complex systems using Modelica and tool coupling.* In *5th Modelica Conference*. vol. 2, 485–490.

Lea D. and Marlowe J., 1995. *Interface-based protocol specification of open systems using PSL.* In *European Conference on Object-Oriented Programming*. Springer, 374–398.

Li X.; Lei Y.; Wang W.; Wang W.; and Zhu Y., 2013. *A DSM-based multi-paradigm simulation modeling approach for complex systems.* In *Simulation Conference (WSC), 2013 Winter*. IEEE, 1179–1190.

Rezaei R.; Chiew T.k.; and Lee S.p., 2013. *A review of interoperability assessment models. Journal of Zhejiang University SCIENCE C*, 14, no. 9, 663–681.

Szyperski C., 1996. *Independently extensible systems-software engineering potential and challenges. Australian Computer Science Communications*, 18, 203–212.

Tripakis S., 2015. *Bridging the semantic gap between heterogeneous modeling formalisms and FMI.* In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 International Conference on*. IEEE, 60–69.

Yoo S. and Choi K., 1997. *Optimistic timed HW-SW cosimulation.* In *in Proc. of APCHDL97*. Citeseer.