OMSimulator – Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP

Lennart Ochel[1] Robert Braun[1] Bernhard Thiele[2] Adeel Asghar[1] Lena Buffoni[1] Magnus Eek[3] Peter Fritzson[1] Dag Fritzson[4] Sune Horkeby[5] Robert Hällquist[3] Åke Kinnander[5] Arunkumar Palanisamy[1] Adrian Pop[1] Martin Sjölund[1]

[1]PELAB – Programming Environment Lab, Dept. of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden, {lennart.ochel, robert.braun}@liu.se
[2]Institute of System Dynamics and Control, German Aerospace Center (DLR), 82234 Weßling, Germany, bernhard.thiele@dlr.de
[3]Saab AB, Bröderna Ugglas gata, SE-582 54 Linköping, Sweden
[4]SKF AB, SE-415 50 Göteborg, Sweden
[5]Siemens Turbomachinery AB, Slottsvägen, SE-612 31 Finspång, Sweden

Abstract

OMSimulator is an FMI-based co-simulation tool and recent addition to the OpenModelica tool suite. It supports large-scale simulation and virtual prototyping using models from multiple sources utilizing the FMI standard. It is integrated into OpenModelica but also available stand-alone, i.e., without dependencies to Modelica-specific models or technology. OMSimulator provides an industrial-strength open-source FMI-based modelling and simulation tool. Input/output ports of FMUs can be connected, ports can be grouped to buses, FMUs can be parameterized and composed, and composite models can be exported according to the (preliminary) SSP (System Structure and Parameterization) standard. Efficient FMI-based simulation is provided for both model-exchange and co-simulation. TLM-based tool connection is provided for a range of applications, e.g., Adams, Simulink, Beast, Dymola, and OpenModelica. Moreover, optional TLM (Transmission Line Modelling) domain-specific connectors are also supported, providing additional numerical stability to co-simulation. An external API is available for use from other tools and scripting languages such as Python and Lua. The paper gives an overview of the tool functionality, compares with related work, and presents experience from industrial usage.

Keywords: FMI, FMU, SSP, modelling, simulation, co-simulation, composite

1 Introduction

The use of virtual prototyping methods in product development has become an indispensable tool to manage the complexity of competitive modern products and industrial processes. Modelling the dynamic behaviour of such products and processes often requires considering systems that are composed of physical subsystems (usually from different physical domains) together with computing and networking. The Modelica language, which allows integrating discrete-time dynamics (e.g., control software) and continuous-time dynamics (process behaviour), is well suited for this task.

However, a frequent problem in larger industrial projects is that although component-level models are available, it is a big hurdle to integrate them into larger system simulations. This is because different development groups and disciplines, e.g., electrical, mechanical, hydraulic, and software, often use their own approaches and special purpose tools for modelling and simulation.

To improve the interoperability of behavioural models, the MODELISAR project (MODELISAR Consortium, 2011), developed the Functional Mock-up Interface (FMI) as a standardized exchange format for behavioural models. Figure 1 illustrates the basic concept: Model components are exported as Functional Mock-up Units (FMUs) from their respective discipline specific tool, another simulator tool can import the FMUs and integrate them into a Functional Mock-up using a suitable master algorithm for coupling the individual units. In October 2014, the improved version FMI 2.0 was released to the public (FMI development group, 2014).

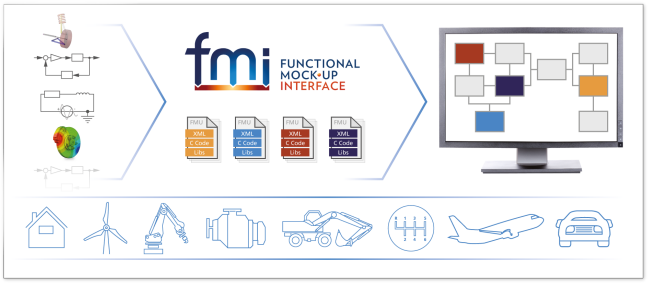

Figure 1. Model integration using FMI (source: https://www.fmi-standard.org/).

The motivation behind FMI is easily understood, however, coupling different simulator codes is a major challenge and an active research area. Modular simulation

of a global system by coupling different simulator codes may easily result in an unstable integration or may require proceeding in prohibitively small time steps (Schierz and Arnold, 2012). Successful co-simulation needs:

- A suitable module interface (this is what FMI standardizes) and
- A suitable master algorithm for coupling the modules (not standardized in FMI).

In previous work, Transmission Line Modelling (TLM) was integrated as one possible approach to co-simulation in OpenModelica (Siemers et al., 2006), also considered as one approach to gain speed-up by simulation parallelization during the RTSIM project (Sjölund et al., 2010; Sjölund, 2015), however, this was not based on FMI. Additional difficulties arise if discrete-time models (e.g., control software) are included within a co-simulation setup (hybrid co-simulation). With regards to hybrid co-simulation, the latest FMI 2.0 standard was shown to have deficiencies and different proposals to amend these deficiencies were discussed, e.g., (Broman et al., 2013; Cremona et al., 2016; Tavella et al., 2016; Cremona et al., 2017).

This paper describes an industrial-strength co-simulation approach. First, it discusses FMI for co-simulation in general and then it introduces the OMSimulator tool framework in Section 3. Based on that, the graphical user interface is outlined in Section 4 and some industrial applications are discussed in Section 5.

2 FMI for Co-Simulation

The FMI 2.0 standard defines two interfaces (FMI development group, 2014, p. 4):

- FMI for Model Exchange (FMI-ME): The intention is that a modelling environment can generate C code of a dynamic system model that can be utilized by other modelling and simulation environments.
- FMI for Co-Simulation (FMI-CS): The intention is to provide an interface standard for coupling simulation tools in a co-simulation environment.

The two interfaces share common parts and concepts, in particular:

- FMI C-application programming interface (API): All computations are evaluated by calling standardized C-functions.
- FMI Extensible Markup Language (XML) description schema: The schema describes the structure and content of an XML file (named modelDescription.xml) generated by the modelling environment which exports an FMU. This modelDescription.xml file contains the definition of all variables and other structural information of an FMU in a standardized form.
- An FMU is delivered as a zip file which contains the XML description file, the code that provides the C-API either in binary form as shared library or as source code, as well as potential additional resources, e.g., tables, model icon, and documentation.

Basically, FMI-ME differs from FMI-CS in that it requires the importing tool to provide a numerical solver for simulating the FMU. Such solvers require vectors for states, derivatives and zero-crossing functions which are exposed by the FMI-ME API. By contrast, FMI-CS does not require the importing tool to provide a numerical solver. Instead, all required solvers are embedded within the FMI-CS and the related information is not exposed by the FMI-CS API.

2.1 FMI-based Co-Simulation

An FMI-based composite model for co-simulation can be constructed with both co-simulation and model-exchange FMUs. The building blocks determine certain constraints of the composite model structure. A straightforward derived structure from the FMI specification is given in Table 1.

Table 1. Overview of co-simulation building blocks.

solver	**components**
master algorithm	co-simulation units • CS-FMU • integrator + set of ME-FMUs
integration method	set of ME-FMUs

The master algorithm forces the so-called global time steps, which are used to exchange information between co-simulation units. Each co-simulation unit takes its own local time steps to reach the next forced global time step.

A co-simulation unit can be composed of a set of ME-FMUs. In this case, these ME-FMUs can communicate with a higher exchange rate than the global time step, basically at each local time step.

2.1.1 Initialization

Initialization must be performed within a dedicated initialization mode. A consistent initial state is computed based on the dependency information provided by the FMUs (optional FMI feature) and the actual connections between the FMUs. First, all parameters will be set to either predefined values or explicitly overwritten by the user's input. The same applies to start values, which might be crucial for internal nonlinear systems and external algebraic loops. After that, all the information is propagated based on the dependency information.

2.1.2 Simulation

The continuous simulation is performed by a master-algorithm which synchronises all co-simulation units and exchanges information between them based on internal

DOI 10.3384/ecp1915769

output-input dependencies and external input-output dependencies. The continuous simulation gets interrupted if an event is detected or the final simulation time is reached.

2.1.3 Event handling

The discrete event simulation takes place if discrete changes are detected. Co-simulation FMUs cannot expose internal events, which means that only discrete changes in output variables at communication time points can be detected. In that case, the changes are propagated and a new consistent model state is computed. This might be an iterative process in case of algebraic loops.

The situation for model-exchange FMUs is a bit different. A set of connected exchange-FMUs are simulated using a shared solver and events can be processed and communicated within this set of FMUs directly when they occur.

2.2 Numerically Stable Co-Simulation

Co-simulation requires different parts of the complete model to be solved separately by isolated solvers. This will inevitably delay the interchanged variables to the next communication step. Such delays may affect numerical stability and simulation accuracy.

In many cases, a master algorithm with fixed communication step size is used and the step size is reduced until the results appear to be stable for the given problem. This is performance consuming and can only ensure stability in the observed working points. More sophisticated solutions include adaptive communication step-size (Schierz et al., 2012) or relaxation techniques (Schweizer et al., 2016). Such methods typically rely on rollback mechanisms, which are often not available (state serialization in FMUs is optional).

One technique that addresses this issue is TLM (Krus, 2011). Every physical element has a finite information propagation speed. By mapping the physically motivated delays to the communication points in the model, artificial time delays can be avoided. As a result, the stability properties of the simulation model will reflect the stability properties of the physical system it represents. In other words, the separation into different solvers will not affect the numerical stability of the complete model. The TLM implementation in OMSimulator is based on previous work by SKF (Siemers et al., 2009; Fritzson et al., 2018). The boundary equations for a TLM connection are shown in Equation 1 and 2:

$$e_1(t) = e_2(t-\Delta t) + Z_c\left[f_1(t) + f_2(t-\Delta t)\right] \quad (1)$$

$$e_2(t) = e_1(t-\Delta t) + Z_c\left[f_2(t) + f_1(t-\Delta t)\right] \quad (2)$$

e_1, e_2: effort variables
f_1, f_2: flow variables
Z_c: characteristic impedance
Δt: time delay

It can be noted that the effort variable on one side of the connection is always independent of variables on the other side within a (usually small) time frame of Δt, during which solvers on both sides can work independently.

With FMI for co-simulation, sub-models can only exchange variables at communication time points. This induces sampling errors, which greatly reduces the benefits of TLM. OMSimulator addresses this by supporting interpolation, either by sending derivatives of the input signals or by providing the sub-models with interpolation tables (Braun et al., 2017b). Based on the assumption that sampling errors arise from aliasing, a related solution could be to use anti-aliasing filters (Benedikt et al., 2013; Drenth, 2017). Another solution based on increasing communication step size using context-based extrapolation was proposed by (Khaled et al., 2014). Both interpolation and anti-aliasing features would greatly benefit from callback functions for writing intermediate outputs and requesting intermediate inputs. This improvement has been suggested to the FMI design group.

3 OMSimulator Tool Framework

OMSimulator is a unified co-simulation tool that supports FMI 2.0 for model exchange and co-simulation. One of its unique features is the support of TLM for numerically stable co-simulation. Simulations can be performed as soft real-time or offline simulations.

3.1 Main Framework Aspects

OMSimulator is developed as a standalone open-source simulation library with a rich C-API. The integration into the OpenModelica graphical editor OMEdit demonstrates how the C-API can be utilized for providing an intuitive (graphical) user experience. Additionally, OMSimulator provides a command-line interface (CLI) and scripting interfaces for Python and Lua. These different interfaces can be used to integrate OMSimulator into third-party tools and specialized applications, e.g. flight simulators and optimization applications.

The open-source implementation enables research on various co-simulation questions, e.g. dependency-graph-based master algorithms for parallel and multi-rate execution of FMI components.

3.2 Simulation Architecture

Composite models are constructed as a tree of certain building blocks. The root node is either a TLM system, weakly-coupled system (WC system), or strongly-coupled system (SC system). The systems differ in the way connections are handled:

- **TLM** systems contain TLM connections, which can basically be considered as physical-motivated delayed connections.
- **Weakly-coupled** systems are used for actual co-simulation. All simulation units run independently

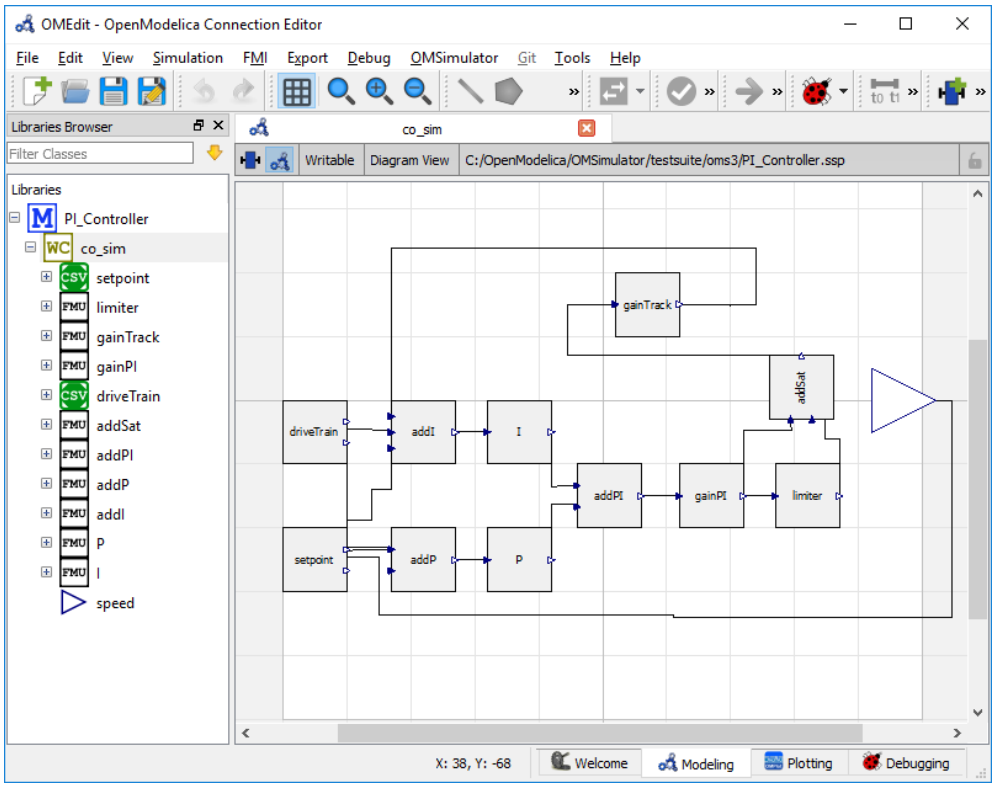

Figure 2. PI controller model created from 9 FMUs and connected to 2 lookup tables for the boundary conditions.

and are synchronized by a master algorithm at certain communication time points.

- **Strongly-coupled** systems are used to wrap-up model exchange FMUs into a co-simulation unit. They share a common solver and use a continuous communication schema.

A system can contain other systems, components (i.e. FMUs or lookup tables), connectors, and buses.

4 Graphical User Interface with Composite Model Editor

A graphical user interface has been developed as an extension to the existing OpenModelica Connection Editor (OMEdit) (Asghar and Tariq, 2010) and the composite model editor presented in (Mengist et al., 2015). OMEdit communicates with OMSimulator through the C-API for visual composite modelling.

4.1 Visual Modelling

The graphical user interface allows the user to create composite models and add systems, components (FMUs, tables and external models), connectors, buses and connections to the model. Each composite model is displayed in the form of a hierarchical tree as shown in the left column of Figure 2.

Each element in the hierarchical tree consists of an icon, diagram and text view except for the top-level model, connectors and buses. The model element does not have an icon view and the connectors and buses are non-editable shapes.

A user can create a connection between two connectors or between two buses. A bus or a TLM bus consists of a list of connectors. When a connection between two buses is made, a bus connection dialog is shown (see Figure 3). The dialog maps the inputs and outputs of the buses automatically. This allows making connections for large systems trivial.

4.2 Simulation and Post Processing

The model needs to be in the instantiated state before performing the simulation. Once the model enters into the instantiation phase the user can set the FMU parameters and start the simulation. The user interface shows the simulation status and progress using the callback functions from the C-API. The simulation results are visualized in the plotting perspective of OMEdit as shown in Figure 4.

5 Industrial Applications and Benchmarks

In this section, several industrial applications are presented.

5.1 Saab Use Case

Analysing and designing sub-systems separately is not enough in modern aircraft development. A competitive product needs to be developed considering the joint behaviour of tightly coupled sub-systems in order to avoid

DOI 10.3384/ecp1915769

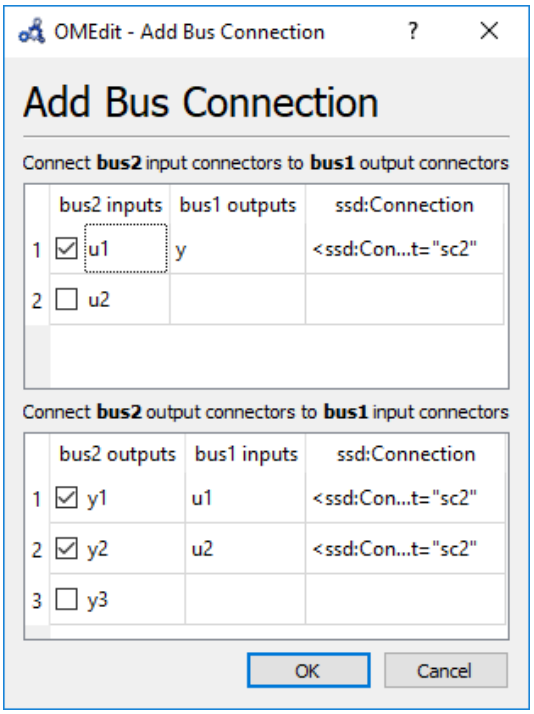

Figure 3. Bus connection.

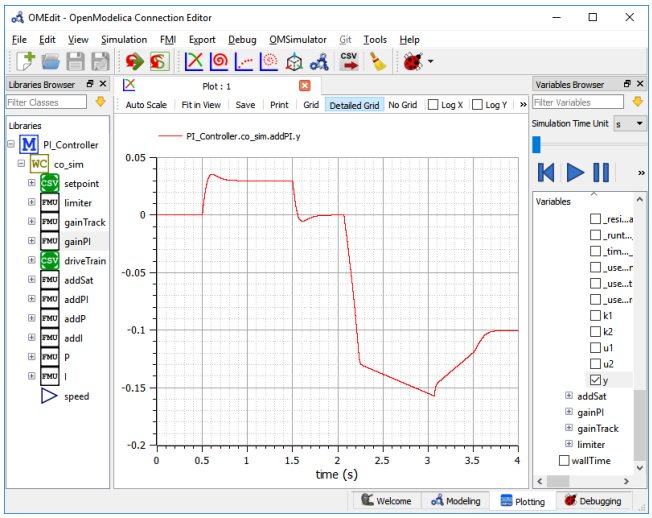

Figure 4. Simulation result.

sub-optimization as well as to achieve the desired high level of aircraft integration. Engineers and researchers, therefore, need to have the means of detailed analysis using coupled simulation models, developed in a wide variety of different domain-specific tools, available on their desktop computers. Scalable, numerically stable, and distributed simulations need to be achieved while preventing tool vendor lock-in effects as well as minimizing licensing costs (Hällqvist et al., 2018).

A detailed aircraft vehicle systems simulator is developed throughout the OpenCPS project. The simulator aims to serve as an industrially relevant platform for testing standardized methods for connecting and simulating models from different tools in the OMSimulator as well as other integrating simulation tools. The aircraft systems simulator is developed in parallel to the OMSimulator, continuously exposing industrial needs and requirements that were not captured during the master simulation engine specification phase (OpenCPS project partners, 2016). An early prototype of the aircraft vehicle systems simulator was presented in (Hällqvist et al., 2017). The simulator was further developed and expanded to enable studies of pilot thermal comfort connected to Environmental Control System (ECS) performance (Hällqvist et al., 2018; Schminder et al., 2018). The latter combines the domains of hardware, software, and human factors modelling. Two different composite models of the same system were created: one using only traditional connections between FMUs, referred to as an FMI composite model, and one with only TLM type connections, referred to as a TLM composite model.

A schematic description of the different included subsystems is presented in Figure 5. The simulator includes an engine model designed to provide the included ECS with air at high temperature and pressure depending on the aircraft boundary conditions. The boundary conditions are expressed by the aircraft operational point along with outputs from the included atmosphere model. In turn, the ECS provides its consumers with conditioned air at the correct mass flow, temperature, and pressure. The specified mass flows, temperatures, and pressures are achieved via a total of five modelled motorized valves controlled by a modelled software, denoted ECS Control in the figure. The included consumers are a thermoregulatory cockpit model, described in detail by Schminder et al. in (Schminder et al., 2016), along with two simple place-holder consumers representing subsystems requiring air cooling and/or pressurization. The cockpit model provides necessary inputs to the included pilot comfort model which incorporates numerous well-established comfort measures into the simulation, such as the Fighter Index of Thermal Stress (Nunneley and Stibley, 1979). The Engine, ECS, and ECS Control models are expressed using the Modelica language whereas the atmosphere, cockpit, and pilot comfort models are developed in Matlab/Simulink (MathWorks). All models are exported as FMUs for co-simulation, the Modelica models using Dymola (Dassault Systemes AB), and the Matlab/Simulink models using the Dassault developed toolbox FMI Kit for Simulink. The Modelica models are all exported with the variable order and variable step solver CVODE (Lawrence Livermore National Laboratory) whereas the Matlab/Simulink models are exported with fixed step solvers.

In (Braun et al., 2017b), different approaches to establishing interoperability between FMI for Co-Simulation and TLM were developed and evaluated. The most suited approach of using callback functions for FMUs to request inputs at the times they are needed is not possible with FMI 2.0. One feasible workaround is to use fine-grained

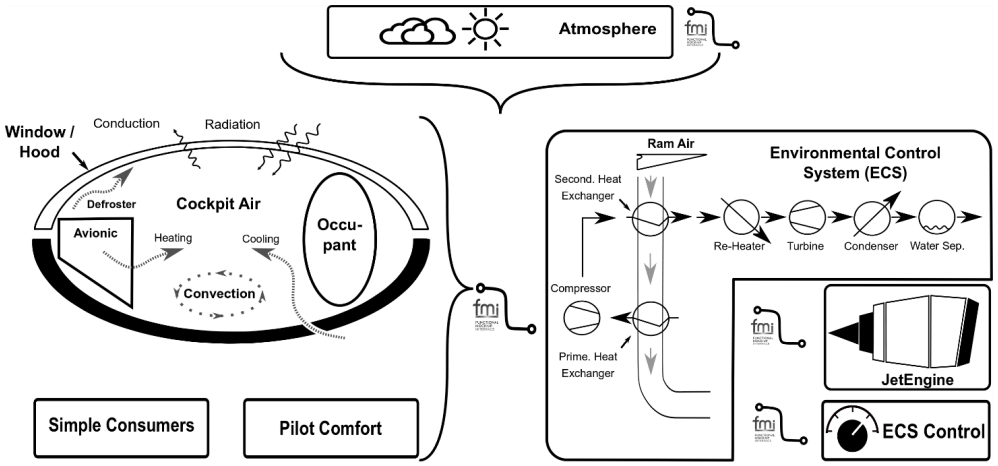

Figure 5. Schematic overview of an aircraft systems simulator comprising detailed sub-system simulation models.

interpolation inside FMUs, see Section 2.2. The mission simulation presented here serves as an industry grade verification test-case for the method of using fine-grained interpolation, in the OMSimulator, to ensure numerical stability during transient conditions.

A subset of the mission boundary conditions are presented in Figure 6, the altitude corresponds to the left-hand side y-axis and the Mach number to the right-hand side y-axis. The ECS consumer supply pressure is plotted for simulations using only TLM connections (Red) and using only traditional native FMI connections (Black) in Figure 7. The results are similar and the discrepancies are likely a result of aliasing effects resulting from a slightly too long communication interval in the native FMI simulation. The main difference between the presented simulations is that all included FMUs are executed in parallel, using physically motivated delays, for the TLM composite model. In contrast to the native FMI zero-order-hold sampling resulting in constant input to FMUs during each master step, the TLM solution guarantees the availability of interpolated input data at the discretion of each FMU's internal solver. For this particular composite model example, the TLM parallelization does not decrease the simulation execution time compared to the native FMI simulation. The main reason for this is that the composite model is not well structured from a parallelization perspective, and thus the FMU representing the ECS physical system clearly dominates the computational effort. In addition, the native FMI simulation is tuned to use the largest communication interval possible challenging the numerical stability of the master simulation, whereas the communication interval in the TLM simulation is based on the real physics and does not compromise numerical stability. Further up-scaling by adding FMUs for other aircraft subsystems, such as a fuel system, a hydraulic system, and an auxiliary power unit, would reveal the scalability benefits, in terms of execution time, of the TLM solution.

The presented use case demonstrates the OMSimulator as an industrially relevant open-source alternative or complements to existing FMI-supporting master simulation tools in aircraft vehicle systems applications. Combining the TLM technique with the more traditional method of simulating coupled FMUs is a most promising and flexible approach for scalable, numerically stable and accurate, distributed simulation. The use case shows that combining models from multiple modelling and simulation domains, from both industry and academia, is feasible using the OMSimulator. In (Hällqvist et al., 2018) and (Schminder et al., 2018), the focus is placed on studies relating ECS performance to pilot thermal comfort. Other possible areas of application are various optimization studies, e.g., minimizing the engine air consumed by the ECS while maximizing pilot comfort.

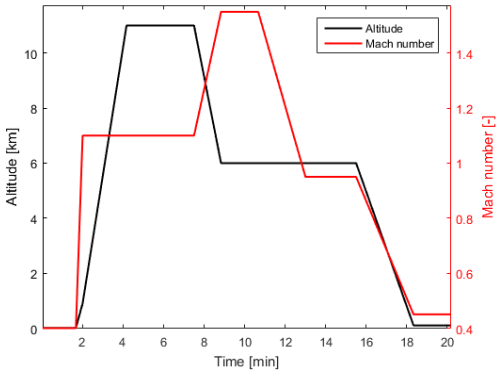

Figure 6. A subset of the simulation boundary conditions, Altitude and Mach number.

5.2 Energy Demonstrator

Driven by the need to limit global warming, the energy systems worldwide are in a phase of expanding renewable energy as an alternative to conventional power plants.

The design and control of combined cycle power plants are expected to become increasingly more important as a method of balancing the electric networks with a large share of renewable energy input. This demonstrator is mo-

DOI 10.3384/ecp1915769

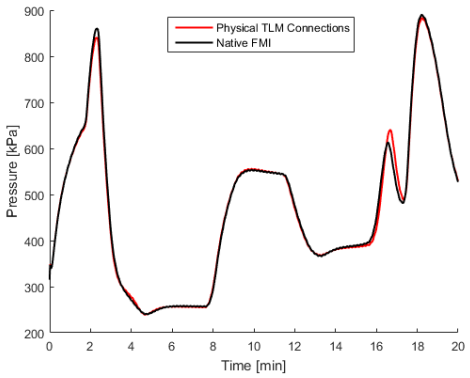

Figure 7. ECS supply pressure to included fuel system model. Results from the TLM Composit model simulation is depictd as red and the Native FMI Composite model simulation as black.

tivated by the need to enable suppliers, in an early design phase, to test the complete functionality by utilizing well-verified models from different sources, without having to convert all models to run by the same tool. This is needed for the entire electrical grid.

The project goal of this joint energy demonstrator was to combine FMUs from four different suppliers, to show that each supplier's verified knowledge, expressed by their FMU, could be used for design, and transient analysis of a power plant.

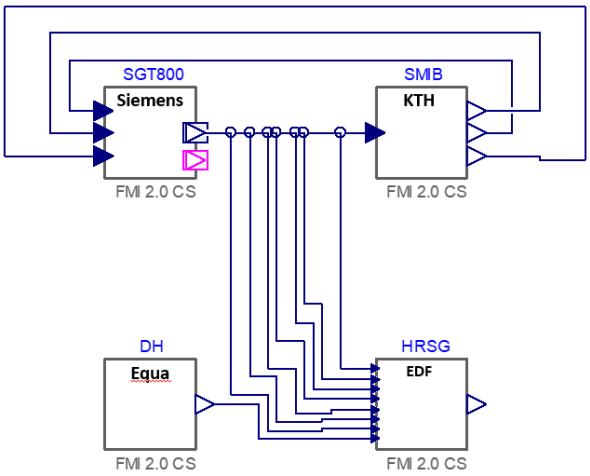

Figure 8. Energy demonstrator of a combined cycle power station with detailed accurate models (total over 30000 equations) from different suppliers, provided as FMUs.

The power plant in Figure 8 is a combined cycle plant (CCPP) with steam extraction to a district heating system. The FMUs are a gas turbine (GT) supplying flue gases to a heat recovery steam generator (HRSG) that supplies steam to a district heating system (DH). The GT shaft drives a generator connected to a large utility network, the model named SMIB. The HRSG also supplies steam to a steam turbine that is included in the HRSG model.

Following entities supplies FMUs for the CCPP:

1. Siemens Industrial Turbomachinery AB supplies the GT
2. KTH supplies the net model
3. EDF supplies the HRSG with ST
4. Equa AB supplies the DH

The simulation results of the generator power during GT start-up from the model shown in Figure 8 depend on communication interval and error tolerance. To achieve accurate results, the simulation settings need to be tightened up which increases the simulation time dramatically. This encouraged us to develop further advanced simulation technologies, such as a master-algorithm with variable step size and input extrapolation based on output derivative information.

OMSimulator has the capability for early multi-domain simulations in the design and configuration phase but also supporting behaviour control in the operational and recycling phase and support closed loops for a sustainable environment.

With this new technology and with the promising test results we will be able to support the vision of sustainable zero emission power plants with optimized solutions and also bridge technologies from different partners.

5.3 SKF 3D Mechanical Demonstrator

Models of 3D mechanics typically contain stiff equations and short time constants. This makes them especially sensitive to delayed variables and thereby poses an interesting challenge for co-simulation. To demonstrate the stability benefits of TLM, a model of a hydraulic crane with two actuators was developed, see Figure 9. The intention is to simulate a model of a roller bearing from SKF together with the surrounding system for achieving accurate boundary conditions. SKF is one of the world's largest suppliers of bearings and has a great interest in simulating their bearing models together with models from customers. This model constitutes a typical scenario, where a system model developed by a customer is connected to a bearing model developed by the supplier. An early prototype of the demonstrator was presented in (Braun et al., 2017a). Table 2 shows an overview of the sub-models in the composite model. All mechanical bodies are modelled in Dymola and exported as FMUs. The crane arms are connected through a roller bearing modelled in SKF BEAST (Fritzson et al., 2014, 2018). The crane mechanics is modelled using rigid bodies, while the bearing model contains flexible bodies and contact mechanics. Motion is controlled by a hydraulic system modelled in Hopsan, a system simulation tool specialized for hydraulic and mechatronic systems developed by Linköping University (Axin et al., 2010). Experiments show that the model works well with FMI for model exchange. With FMI for co-simulation, either callback functions or fine-grained interpolation (see section 2.2) are required to achieve stable

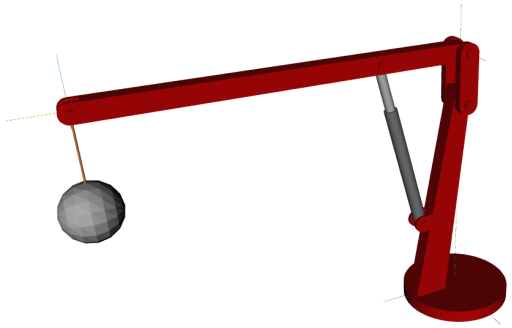

Figure 9. The SKF crane demonstrator model is used to verify stability in 3D connections.

results. When using sampled inputs with zero-order hold, stability cannot be achieved even when using a step-size 1000 times smaller than the other methods. Results and performance cannot be compared to a monolithic implementation because there is no tool capable of simulating all three parts of the model: the bearing, the crane and the hydraulic system. Nevertheless, results are fully realistic. The motion agrees well with simplified models. Static forces and torques all have correct magnitudes. No unaccountable phenomena have been observed.

Table 2. Overview of the different sub-models in the SKF demonstrator model.

Sub-model	Tool
Boom	FMU (Dymola)
Jib	FMU (Dymola)
Load	FMU (Dymola)
Piston	FMU (Dymola)
Bearing	BEAST
Hydraulics	Hopsan
Controller	Hopsan

6 Related Work

Table 3 compares related tools and libraries that also support similar co-simulation functionality. There are both commercial and free-of-charge solutions available with different licences. All tools support both model-exchange and co-simulation FMUs. PyFMI is the only tool which does not include built-in support for lookup tables. This feature is quite handy, but not critical because it can be isolated and solved by dedicated FMUs.

Except for Simulink, all tools support handling of algebraic loops. In Simulink, however, it is not possible to execute such composite models. Introducing a delayed signal can circumvent this issue, but is not considered as an appropriate solution since it introduces unintended dynamics.

All the tools except FMI Composer (Modelon), provide some kind of scripting interface. DACCOSIM (Virginie et al., 2015), Simulink and Dymola have proprietary solutions and OMSimulator, PySimulator (Pfeiffer et al., 2012; Asghar et al., 2015), FMI Go! (Lacoursière and Härdin, 2017), and PyFMI (Christian et al., 2016) are based on open scripting languages.

OMSimulator uses the upcoming SSP standard as an exchange format for composite models, as well as FMI Go!, and FMI Composer.

7 Conclusions

OMSimulator 2.0 is part of the OpenModelica 1.13.0 release and also available as a standalone application. It provides the following functionality. It supports both FMI variants, i.e. model-exchange and co-simulation. It supports also the TLM technique to decouple co-simulation units and potentially stabilize the simulation. TLM connections enable direct tool-coupling as well, e.g. with Adams, Beast, and Simulink.

The OpenModelica graphical editor OMEdit is connected to OMSimulator via a C-API and provides a rich user experience.

The SSP standard, which is still under development, is supported as an early prototype to enable exchanging models with an open and independent standard. As an extension to the current SSP version, signal grouping and bus connections are supported and integrated into the SSP using annotations.

Compared to other tools, OMSimulator has outstanding features like TLM and SSP support. The open-source implementation facilitates use by academics and also in industry. It can be used as a research platform for co-simulation.

Acknowledgements

This work has been supported by Vinnova in the ITEA OPENCPS, and EMPHYSIS projects and in the Vinnova RTISIM project. Support from the Swedish Government has been received from the ELLIIT project. The OpenModelica development is supported by the Open Source Modelica Consortium. Many students, researchers, and engineers have contributed to the OpenModelica system. There is not room here to mention all these people, but we gratefully acknowledge their contributions.

References

Adeel Asghar, Andreas Pfeiffer, Arunkumar Palanisamy, Alachew Mengist, Martin Sjölund, Adrian Pop, and Peter Fritzson. Automatic regression testing of simulation models and concept for simulation of connected FMUs in PySimulator. In Fritzson and Elmqvist (2015). doi:10.3384/ecp15118671.

Syed Adeel Asghar and Sonia Tariq. Design and implementation of a user friendly OpenModelica graphical connection editor. Master's thesis, Linköping University, Department of Computer and Information Science, December 2010. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-65864`.

Table 3. Comparison of related tools.

	OMSimulator	**DACCOSIM**	**Simulink**	**PyFMI**
Commercial	No	No	Yes	No
Open-source	OSMC-PL, GPL	AGPL2	No	LGPL
Lookup Table	Yes	Yes	Yes	No
Alg. Loops	Yes	Yes	No	Yes
Scripting	Python, Lua	proprietary	proprietary	Python
GUI	Yes	Yes	Yes	No
SSP	Yes	No	No	No
platform	Linux/Win/macOS	Linux/Win	Linux/Win/macOS	Linux/Win/macOS

	Dymola	**PySimulator**	**FMI Go!**	**FMI Composer**
Commercial	Yes	No	No	Yes
Open-source	No	BSD	MIT	No
Lookup Table	Yes	Yes	Yes	Yes
Alg. Loops	Yes	Yes	Yes	Yes
Scripting	proprietary	Python	Go	No
GUI	Yes	Yes	No	Yes
SSP	No	No	Yes	Yes
platform	Linux/Win	Linux/Win	Linux/Win/macOS	Linux/Win/macOS

Mikael Axin, Robert Braun, Alessandro Dell'Amico, Björn Eriksson, Peter Nordin, Karl Pettersson, Ingo Staack, and Petter Krus. Next generation simulation software using transmission line elements. In *Fluid Power and Motion Control*, Bath, England, September 2010.

Martin Benedikt, Daniel Watzenig, and Anton Hofer. Modelling and analysis of the non-iterative coupling process for co-simulation. *Mathematical and Computer Modelling of Dynamical Systems*, 19(5):451–470, 2013. doi:10.1080/13873954.2013.784340.

Robert Braun, Adeel Asghar, Adrian Pop, and Dag Fritzson. An open-source framework for efficient co-simulation of fluid power systems. In *Proceedings of 15th Scandinavian International Conference on Fluid Power, June 7-9, 2017*, number 144, pages 393–400, Linköping, Sweden, June 2017a. Linköping University Electronic Press, Linköpings universitet.

Robert Braun, Robert Hällqvist, and Dag Fritzon. TLM-based Asynchronous Co-simulation with the Functional Mockup Interface. In *IUTAM Symposium on Solver Coupling and Co-Simulation*, Darmstadt, Germany, September 2017b.

David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. Determinate composition of fmus for co-simulation. Technical Report UCB/EECS-2013-153, EECS Department, University of California, Berkeley, Aug 2013. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-153.html`.

Andersson Christian, Åkesson Johan, and Führer Claus. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface. Technical Report 2, Centre for Mathematical Sciences, Lund University, 2016. URL `https://lup.lub.lu.se/search/publication/961a50eb-e4a8-43bc-80ac-d467eef26193`.

Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A. Lee. Fide: An fmi integrated development environment. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 1759–1766, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3739-7. doi:10.1145/2851613.2851677.

Fabio Cremona, Marten Lohstroh, David Broman, Edward A. Lee, Michael Masin, and Stavros Tripakis. Hybrid co-simulation: it's about time. *Software & Systems Modeling*, November 2017. ISSN 1619-1374. doi:10.1007/s10270-017-0633-6.

Dassault Systemes AB. Dymola. URL `https://www.3ds.com/products-services/catia/products/dymola/`.

Edo Drenth. Method and system for control and co-simulation of physical systems, March 2 2017. US Patent App. 15/232,261.

FMI development group. Functional Mock-up Interface for Model Exchange and Co-Simulation v2.0. Modelica Association Project "FMI", October 2014. URL `https://www.fmi-standard.org/`. Standard Specification.

Dag Fritzson, Lars-Erik Stacke, and Jens Anders. Dynamic simulation–building knowledge in product development. *Evolution*, 1, 2014.

Dag Fritzson, Robert Braun, and Jan Hartford. Composite modelling in 3-d mechanics utilizing transmission line modelling (tlm) and functional mock-up interface (fmi). 2018.

Peter Fritzson and Hilding Elmqvist, editors. *Proceedings of the 11th International Modelica Conference*, September 2015. Modelica Association and Linköping University Electronic Press. doi:10.3384/ecp15118.

Robert Hällqvist, Rober Braun, and Petter Krus. Early Insights on FMI-based Co-Simulation of Aircraft Vehicle Systems. In

Proceedings of the 15th Scandinavian International Conference on Fluid Power, Linköping, Sweden, 2017.

Robert Hällqvist, Jörg Schminder, Magnus Eek, Robert Braun, Roland Gårdhagen, and Peter Krus. A novel FMI and TLM-based simulator for detailed studies of thermal pilot comfort. In *Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences (ICAS)*, Belo Horizonte, Brazil, 2018.

Abir Ben Khaled, Laurent Duval, Mohamed El Mongi Ben Gaïd, and Daniel Simon. Context-based polynomial extrapolation and slackened synchronization for fast multi-core simulation using fmi. In *International Modelica Conference*, pages 225–234. Linköping University Electronic Press, 2014.

Petter Krus. Robust modelling using bi-lateral delay lines for high speed simulation of complex systems. In *DINAME 2011: 14th International Symposium on Dynamic Problems in Mechanics*, 2011. URL http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-67897. Invited conference contribution.

Claude Lacoursière and Tomas Härdin. Fmi Go! A Simulation runtime environment with a client server architecture over multiple protocols. In *12th Int. Modelica Conference*, Prague, Czech Republic, 2017. URL https://www.modelica.org/events/modelica2017/proceedings/html/submissions/ecp17132653_LacoursiereHardin.pdf.

Lawrence Livermore National Laboratory. SUNDIALS: SUite of Nonlinear and Differential/ALgebraic Equation Solvers. URL https://computation.llnl.gov/projects/sundials/cvode.

MathWorks. Simulink. URL https://www.mathworks.com/products/simulink.html.

Alachew Mengist, Adeel Asghar, Adrian Pop, Peter Fritzson, Willi Braun, Alexander Siemers, and Dag Fritzson. An open-source graphical composite modeling editor and simulation tool based on fmi and tlm co-simulation. In Fritzson and Elmqvist (2015). doi:10.3384/ecp15118181.

MODELISAR Consortium. MODELISAR - From System Modeling to S/W running on the Vehicle, 2011. URL https://itea3.org/project/modelisar.html.

Modelon. FMI COMPOSER. URL https://www.modelon.com/products-services/modelon-deployment-suite/fmi-composer/.

Sarah Nunneley and Richard Stibley. Fighter Index of Thermal Stress: Development of Interim Guidance for Hot-Weather USAF Operations. *Journal of the American Society of Heating and Ventilating Engineers*, 50:639–642, 1979.

OpenCPS project partners. FMI Master Simulation Tool Requirement Specification, December 2016. URL https://itea3.org/project/opencps.html.

Andreas Pfeiffer, Matthias Hellerer, Stefan Hartweg, Martin Otter, and Matthias Reiner. PySimulator – A Simulation and Analysis Environment in Python with Plugin Infrastructure. In *9th Int. Modelica Conference*, Munich, Germany, 2012. URL http://www.ep.liu.se/ecp/076/053/ecp12076053.pdf.

Tom Schierz and Martin Arnold. Stabilized overlapping modular time integration of coupled differential-algebraic equations. *Applied Numerical Mathematics*, 62(10):1491 – 1502, 2012. ISSN 0168-9274. doi:10.1016/j.apnum.2012.06.020.

Tom Schierz, Martin Arnold, and Christoph Clauß. Co-simulation with communication step size control in an FMI compatible master algorithmnak. In *9th Int. Modelica Conference, Munich, Germany*, pages 205–214, 2012.

Jörg Schminder, Roland Gårdhagen, Elias Nilsson, Karl Storck, and Matts Karlsson. Development of a Cockpit-Pilot Model for Thermal Comfort Optimization During Long-Mission Flight. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, 2016.

Jörg Schminder, Robert Hällqvist, Magnus Eek, and Roland Gårdhagen. Pilot performance and heat stress assessment support using a cockpit thermoregulatory simulation model. In *Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences (ICAS)*, Belo Horizonte, Brazil, 2018.

Bernhard Schweizer, Daixing Lu, and Pu Li. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. *Multibody System Dynamics*, 36(1):1–36, 2016. ISSN 1573-272X. doi:10.1007/s11044-015-9464-9.

Alexander Siemers, Dag Fritzson, and Peter Fritzson. Meta-Modeling for Multi-Physics Co-Simulation applied for Open-Modelica. In *International Congress on Methodologies for Emerging Technologies in Automation (ANIPLA2006)*, Rome, Italy, November 13–15 2006.

Alexander Siemers, Dag Fritzson, and Iakov Nakhimovski. General meta-model based co-simulations applied to mechanical systems. *Simulation Modelling Practice And Theory*, 17(4): 612–624, 2009. doi:doi:10.1016/j.simpat.2008.10.006.

Martin Sjölund. *Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models*. Doctoral thesis No 1664, Linköping University, Department of Computer and Information Science, 2015.

Martin Sjölund, Robert Braun, Peter Fritzson, and Petter Krus. Towards efficient distributed simulation in modelica using transmission line modeling. In *3rd International Workshop on Equation-Based Object-Oriented Languages and Tools.*, Oslo, Norway, October 2010. URL http://www.ep.liu.se/ecp/047/.

J. P. Tavella, M. Caujolle, S. Vialle, C. Dad, C. Tan, G. Plessis, M. Schumann, A. Cuccuru, and S. Revol. Toward an accurate and fast hybrid multi-simulation with the fmi-cs standard. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–5, September 2016. doi:10.1109/ETFA.2016.7733616.

Galtier Virginie, Vialle Stephane, Dad Cherifa, Jean-Philippe Tavella, Lam-Yee-Mui Jean-Philippe, and Plessis Gilles. Fmi-based distributed multi-simulation with daccosim. pages 39–46, 2015. URL http://dl.acm.org/citation.cfm?id=2872965.2872971.