Early Insights on FMI-based Co-Simulation of Aircraft Vehicle Systems

Robert Hallqvist[*], Robert Braun[**], and Petter Krus[**]

E-mail: Robert.Hallqvist@saabgroup.com, Robert.Braun@liu.se, Petter.Krus@liu.se
[*]Systems Simulation and Concept Design, Saab Aeronautics, Linköping, Sweden
[**]Department of Fluid and Mechatronic Systems, Linköping University, Linköping, Sweden

Abstract

Modelling and Simulation is extensively used for aircraft vehicle system development at Saab Aeronautics in Linköping, Sweden. There is an increased desire to simulate interacting sub-systems together in order to reveal, and get an understanding of, the present cross-coupling effects early on in the development cycle of aircraft vehicle systems. The co-simulation methods implemented at Saab require a significant amount of manual effort, resulting in scarcely updated simulation models, and challenges associated with simulation model scalability, etc. The *Functional Mock-up Interface* (FMI) standard is identified as a possible enabler for efficient and standardized export and co-simulation of simulation models developed in a wide variety of tools. However, the ability to export industrially relevant models in a standardized way is merely the first step in simulating the targeted coupled sub-systems. Selecting a platform for efficient simulation of the system under investigation is the next step. Here, a strategy for adapting coupled *Modelica* models of aircraft vehicle systems to TLM-based simulation is presented. An industry-grade application example is developed, implementing this strategy, to be used for preliminary investigation and evaluation of a co-simulation framework supporting the Transmission Line element Method (TLM). This application example comprises a prototype of a small-scale aircraft vehicle systems simulator. Examples of aircraft vehicle systems are environmental control systems, fuel systems, and hydraulic systems. The tightly coupled models included in the application example are developed in Dymola, OpenModelica, and Matlab/Simulink. The application example is implemented in the commercial modelling tool Dymola to provide a reference for a TLM-based master simulation tool, supporting both FMI and TLM. The TLM-based master simulation tool *TLMSimulator* is investigated in terms of model import according to the FMI standard with respect to a specified set of industrial needs and requirements.

Keywords: FMI, TLM, Modelica, Aircraft Vehicle Systems

1 Introduction

Model Based Systems Engineering (MBSE) [1] is an outspoken strategy for aircraft vehicle systems development at Saab Aeronautics. Methods for efficient model simulator integration, as well as robust and fast simulation, are necessary if the benefits of MBSE are to be fully exploited. The *Functional Mock-up Interface* (FMI) standard is a first step towards reducing the overhead associated with connecting models of interacting sub-systems. At the time of writing, approximatly 50 commercial and open source tools support the latest version of the standard, FMI 2.0 [2]. Furthermore, as far as the authors know, none of these tools support asyncronous simulation implementing the *Transmission Line element Method* (TLM) for numerically stable partitioning of simulation models. An open source master tool, here refered to as the *TLMSimulator*, supporting both of these key technologies is under development within the frame of the *OpenCPS* project [3]. Such a tool is predicted to be of great benefit to MBSE in aircraft vehicle systems development. Efficient model simulator integration will allow model developers, software developers, systems engineers, and flight test engineers to set up large multi-purpose simulation environments as well as small-scale simulators tailored for specific studies. An example of such a study is the investigation of how an aircraft's *Environmental Control System*'s (ECS) performance affects the pilot's thermal comfort, both physically and psychologically. Both types of simulation environments are currently being used for aircraft vehicle systems development at Saab Aeronautics. However, the authors foresee that their use could be further increased as a result of incorporating efficient and standardized methods for model simulator integration.

Here, a strategy for adapting coupled *Modelica* models of aircraft vehicle systems for simulation in an FMI-based simulation environment supporting TLM is described. The paper also describes the implementation of this strategy on an aircraft vehicle systems simulator. Reference simulations of this simulator are conducted for evaluation and development of the TLMSimulator.

This paper is structured as follows. First, a brief introduction to the concepts of FMI and TLM are given in section 1.1

and section 1.2 respectively. A subset of the needs and requirements, identified by the OpenCPS project partners, concerning an industrially applicable tool for simulation of tightly coupled aircraft vehicle system models is presented in section 2. The developed application example is described in section 3, starting with the simulator architecture and continuing with descriptions of the modelled sub-systems. The state-of-the-art open source co-simulation framework for co-simulation using both FMI and TLM is briefly described in section 4. The application example is implemented in both the commercially available Modelica tool *Dymola* and the TLMSimulator. The results obtained from application example Dymola simulations are presented in section 5.1. Finally, the concluding remarks are stated in section 6

1.1 Functional Mock-up Interface standard

The FMI standard is a standardization effort commenced in the *MODELISAR* project [4]. The standard specifies a generic format for export of model executables, referred to as *Functional Mock-up Units* (FMUs). It also includes a set of C functions for calling an FMU along with a standardized interface description xml schema. FMI is a standard for export of FMUs for both co-simulation (a suitable numerical solver is included in the exported model) and model exchange (the central solver is implemented in the integrating tool) [2]. The standard is maintained by the *Modelica association* [5] also responsible for maintaining the Modelica modelling language. The Modelica language is an object-oriented and equation-based modelling language especially suited for multi-domain modelling of physical systems. FMI is here seen as an enabler for reducing overhead costs associated with the exchange of models between tools, not only within the confines of a specific company but also with its subcontractors and suppliers.

1.2 Transmission Line element Method

The Transmission Line element Method (TLM) is a mature and well documented technique for numerically stable partitioning of simulation models. Here, a brief introduction to the most relevant aspects of TLM is given. Krus et al. [6] and Braun et al. [7] provide detailed descriptions of the presented method. Two FMUs connected to each other (FMU$_1$ and FMU$_2$) in the TLMSimulator, see section 4, communicate via TLM connections. A thermodynamic TLM connection is more or less a volume which receives information on volume flow from each interfacing FMU. The net volume flow is integrated in the TLM connection rendering a pressure which is passed back to each of the connected FMUs. The relationship between pressures and volume flows at the interface of the two FMUs is described by

$$p_1(t) = Z_c[q_1(t) + q_2(t - \Delta t_{TLM})] + p_2(t - \Delta t_{TLM}) \qquad (1)$$

$$p_2(t) = Z_c[q_2(t) + q_1(t - \Delta t_{TLM})] + p_1(t - \Delta t_{TLM}) \qquad (2)$$

if friction in the transmission line is disregarded. Interfacing variables of FMU$_1$ are p_1 and q_1 and of FMU$_2$ p_2 and q_2. In Equations (1) and (2), neither the pressure nor the volume flow depends on current information provided by the other FMU. Instead, the information necessary is delayed Δt_{TLM} seconds. This time delay corresponds to the time it takes for a wave to propagate through the connection. The characteristic impedance Z_c of Equations (1) and (2) can be expressed as

$$Z_c = \Delta t_{TLM}/C \qquad (3)$$

where

$$C = V/\beta \qquad (4)$$

for incompressible flow fluid connections. In Equation (4), V is the volume of the TLM connection and β the bulk modulus of the fluid passing through the connection. If C and Δt_{TLM} are physically accurate, then no numerical error will be introduced when connecting FMU$_1$ with FMU$_2$ via a TLM connection. Also, the delay Δt_{TLM} provides a clearly defined time window enabling numerically stable distributed simulation.

2 Requirements

A set of high-level functional and non-functional industrial requirements on a master simulation tool have been formulated by the OpenCPS project partners [3]. Saab's contribution to these requirements are derived from the currently implemented processes and methods applied for aircraft vehicle system modeling and simulation activities at Saab Aeronautics [8]. A subset of these requirements, which are relevant for this study, has been extracted and is shown in Table 1. These high-level requirements are used to guide the application example and TLMSimulator development as well as the preliminary evaluation of the TLMSimulator.

3 Application example

A combined set of interconnected aircraft vehicle systems of a generic fighter aircraft together comprise the application example. This application example is referred to as a small-scale systems simulator. Such a simulator is typically used for system development, training, software verification, fault simulation, performance prediction and evaluation, etc. The application example is specifically developed to facilitate an industry-grade platform for development and evaluation of master simulator algorithms such as the TLMSimulator, see section 4. The simulator architecture is schematically visualized in Figure 1. This prototype version includes models of the *Environmental Control System (ECS)* hardware as well as its controlling software, *ECS Control Software* in Figure 1. A model importing the relevant *Boundary Conditions* is also included. This model incorporates all flight mission data necessary for conducting simulations by reading a *<BoundaryConditions>.mat* file specified by the user. Two different subscribers of coolant air are modelled (*Consumer A* and *Consumer B*) and included along with simple *Cockpit* and *Engine* models.

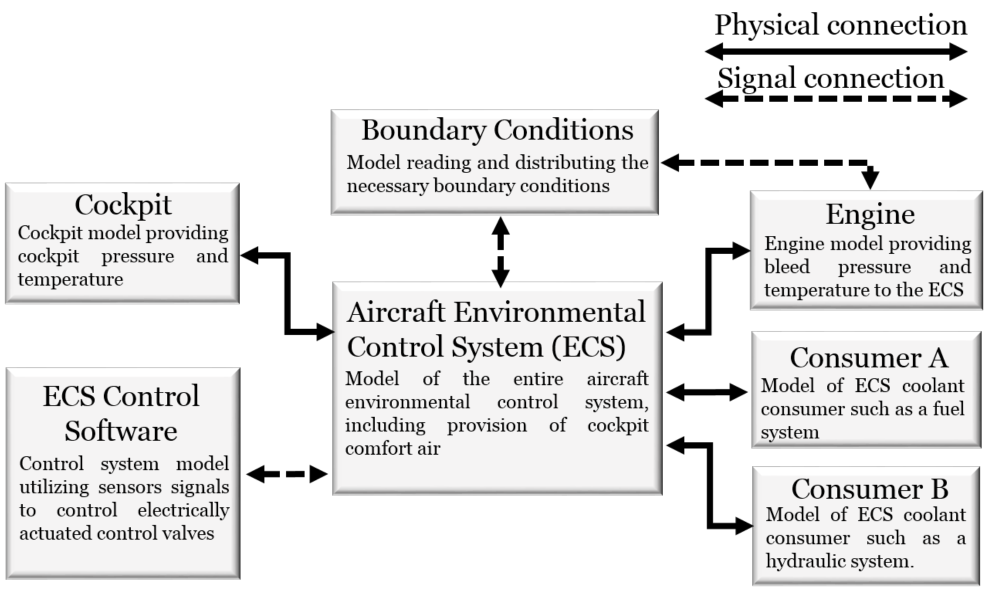
Physical connection
Signal connection
Boundary Conditions
Model reading and distributing the necessary boundary conditions
Cockpit
Cockpit model providing cockpit pressure and temperature
Engine
Engine model providing bleed pressure and temperature to the ECS
Aircraft Environmental Control System (ECS)
Model of the entire aircraft environmental control system, including provision of cockpit comfort air
Consumer A
Model of ECS coolant consumer such as a fuel system
ECS Control Software
Control system model utilizing sensors signals to control electrically actuated control valves
Consumer B
Model of ECS coolant consumer such as a hydraulic system.

Figure 1: Schematic description of application example architecture

Table 1: Subset of requirements on an FMI compliant master simulation tool. The requirements are derived by industrial and academic partners of the OpenCPS project [3]

Number	Name	Description
1	FMU Import	Import of FMUs from tools such as Matlab/Simulink, OpenModelica, and Dymola shall be supported
2	Robustness	Given that all individual FMUs of a simulator configuration are numerically stable, numerical robustness for the simulator configuration as a whole shall be guaranteed
3	Solver configuration	Manual selection of the central solver used for FMUs for Model Exchange shall be available
4	FMU configuration	Support for setting FMU parameters. It must be possible to control start and stop times, solver step size, tolerance, etc.
5	Load simulator configuration	The end user shall be able to load a stored composition of FMUs

The dashed arrows in Figure 1 represent signal connections. The solid lines are physical connections, i.e. information about physical quantities such as volume flow and pressure is passed in its physical form. The latter type of connection is most suited to the TLM technique as it naturally contains a non-negligible time delay, see section 1.2.

3.1 Aircraft Environmental Control System

Aircraft environmental control systems are designed to provide their consumers with pressurized air at the correct mass flow and temperature. An industry-grade ECS model is included in the application example. This particular Modelica model is developed in Dymola using the Saab-developed library *Modelica Fluid Lite* (MFL) [9].

MFL is a stand-alone Modelica library with its own media functions. MFL is not a conventional Modelica library in the sense that components are connected causally strictly via two different types of power port connectors, one volume element connector and one flow element connector. Air with water content in both liquid and gaseous states is selected to be used in the application example as the presence of water greatly affects the ECS' performance limits. In MFL, pressure, mass flow, specific enthalpy and water content are passed between components for such a medium. All MFL components are classified as either volume or flow elements. As an example, one of the most fundamental flow elements in the MFL library is a frictionless pipe. The mass flow through such a pipe is computed via the pressure drop ΔP, provided by the interfacing volume elements, according to

$$\dot{m} = A\sqrt{2\rho\Delta P/Z} \tag{5}$$

where Z is the pressure drop coefficient. The pressure drop coefficient is specified as a component design parameter. The density, ρ, is computed using the pipe mean pressure along with the specific enthalpy and water content also provided by the interfacing volume elements. The most fundamental MFL flow element is the node. The node pressure is calculated using

$$p = \rho R_{specific} T \tag{6}$$

assuming that the present medium is an ideal gas. In contrast to the volume element, the temperature, T, is computed using the specific enthalpy and water content in the node. The density in Equation (3) is computed as

$$\rho = \frac{\int_{t_0}^{t} \dot{m}_{net} dt}{V} \tag{7}$$

where V is the node volume specified as an input parameter.

The top level of the application example ECS model is shown in Figure 3. This specific model is not calibrated to represent any particular aircraft; it is a model of a generic Environmental Control System intended to be of industrially relevant complexity in terms of non-linearities and size. The heart of this system is a bootstrap configuration air-cycle machine which is supplied with conditioned air bled from the aircraft engine. The high bleed air temperature is decreased through the primary heat exchanger via a ram air intake.

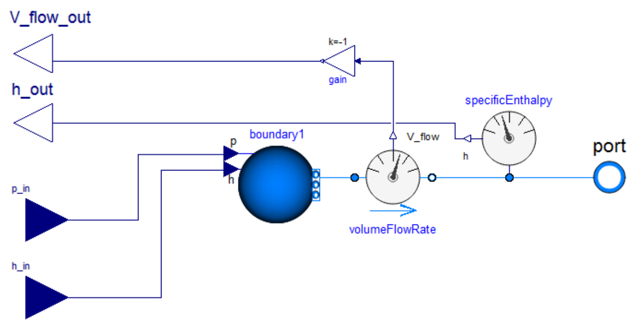

Figure 2: Example of model specifying the causality of models developed in Modelica.Fluid

This air is fed to the air-cycle machine, also referred to as a *cooling pack*, via a motorized control valve that regulates the pack outlet pressure. The cooling pack consists of a compressor, a turbine, multiple heat exchangers, and a water separator. The temperature of the conditioned air exiting the air-cycle machine is controlled to the set points specified by the consumers via two different by-pass branches. Examples of possible consumers of coolant air are the on-board avionics as well as the aircraft's fuel and hydraulic systems. The described ECS configuration is for the most part found in the majority of military fighter aircraft [10].

The ECS inputs/outputs are labelled as either *signal connections* or *physical connections* in Figure 3 just as in Figure 1. The physical connections are where applicable grouped in causal Modelica power port *connectors* [11] that pass information on volume flow, specific enthalpy, water content, and pressure in a predefined direction. Such connectors are beneficial when connecting FMUs of MFL models with one another as the causal connectors are automatically de-grouped

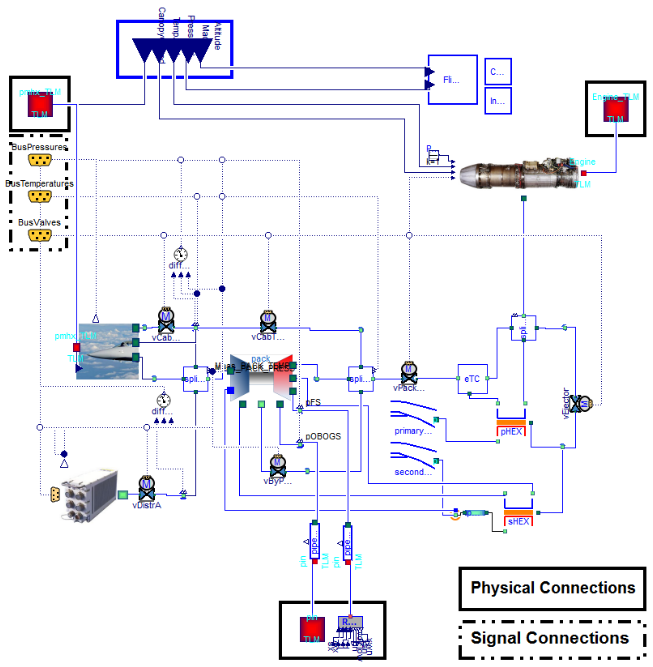

Figure 3: Modelica model of the Environmental Control System hardware

during FMU export from both Dymola and OpenModelica. Casual connectors, however, are not compatible with the acausal stream connectors generally used in Modelica libraries designed for bi-directional flow [12]. *Modelica.Fluid* is an example of such a library. The causality needs to be specified on any acausal interface if a model containing such connectors is to be connected to an MFL model or any FMU. Hirano et al. presents adaptors specifying the causality for signals in the electric and mechanics domains [13]. In Figure 2, a one dimensional thermodynamics domain adaptor for connections between Modelica.Fluid models and MFL volume type components is presented. This particular example is only suited to dry air as only information about volume flow, pressure, and specific enthalpy is passed with fixed causality. However, the adaptor can easily be modified to pass information about present water content as well.

A conventional acausal port from the Modelica.Fluid library is positioned on the right hand side of the adaptor. This port connects directly to any compatible Modelica.Fluid model such as a modelled consumer of ECS-conditioned air. The mass flow passing through the adaptor is a result of the difference in pressure between the pressure source and the connected Modelica.Fluid model. The pressure and specific enthalpy in the pressure source are specified via the two causal connectors on the left hand side of the figure.

A corresponding causal node component is shown in Figure 4. The node component utilizes input information on net volume flow to compute and return the resulting node pressure, see Equation (7). Connecting MFL flow element components with Modelica.Fluid models is easily done if the two adaptors presented in Figure 4 and Figure 2 are combined. The causal node component corresponds to the TLM connection of physical signals in the TLMSimulator. This adaptor is incorporated in the application example to simplify reference simulations in Dymola, see section 5.

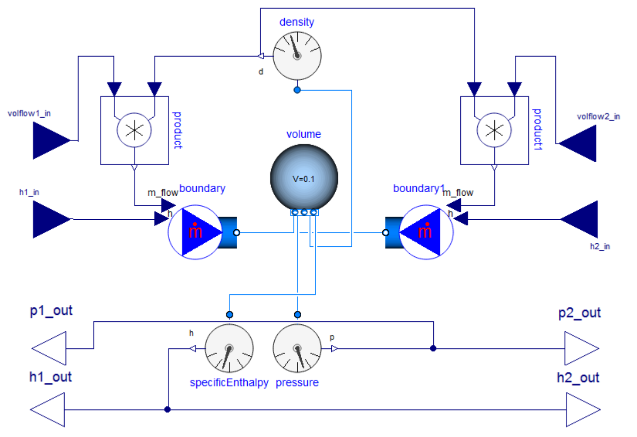

Figure 4: Example of node type model connecting Modelica models with causal interfaces. This adaptor is constructed using components from the Modelica.Fluid library

3.2 ECS Control Software

Here, the ECS Control Software refers to the Environmental Control System controlling software. Two different ECS software models of different level of detail are used in this study. A simple test-stub model is used to verify the interfaces and a slightly more detailed model used for rudimentary control. The latter model is meant to control the ECS' outputs to feasible nominal values during normal operation. Two versions of this more detailed model are developed in parallel, one in *Simulink* and one in *OpenModelica* [14]. The OpenModelica model is developed for reference simulations in any Modelica tool and the Simulink model is developed to provide the coupling between Simulink-generated FMUs and the selected integrating tool. The detailed ECS control software model consists of four proportional controllers. These controllers are empirically tuned to control four of the six motorized ECS valves. They are individually calibrated not considering the present cross-couplings. The ECS valves controlled via the proportional controllers are the valves designed for cooling pack outlet pressure control, cooling pack outlet temperature control, cockpit inlet air temperature control, and avionics cooling power control. The ejector and cockpit comfort air valves are not included. These valves are locked in nominal positions in this early prototype phase. The nominal positions are selected such that reasonable nominal flow, corresponding to normal operating conditions within the ejector flight envelope, are passed through the valves.

3.3 Consumers

Three different consumers are modelled separately and included in the application example: the Cockpit, Consumer A, and Consumer B. Models of the on-board avionics and cockpit avionics are included in the ECS hardware model. Consumers A and B are modelled as generic subscribers of coolant air. Both models consist of a series of pipes, along with a pneumatic self-regulating valve connected to a pressure sink. Such models can be seen as simple representations of an aircraft fuel or hydraulic system providing static and dynamic loads in terms of pressure and mass flow sufficient when the ECS is the system under investigation, during for example performance and fault simulations. In the application example, Consumer A is modelled using components from the Modelica.Fluid library and Consumer B is modelled using components from the MFL library. The cockpit model consists of a volume of representative size. This model provides a feasible inertia to the ECS model. It is separated from the ECS model as a first step towards including a more detailed cockpit model fulfilling the requirements associated with studies of pilot thermal comfort etc. Models of cockpit avionics, the pressure control system, and pilot *Air Ventilated Garments* (AVG) are positioned inside the ECS hardware model. The pressure control system consists of two valves controlling the cockpit pressure to a set point value depending on the aircraft's altitude. The modelled AVGs and cabin avionics are only representative in terms of pressure drop. Such a level of detail is sufficient for most studies where the ECS system is the unit under investigation. However, for in-depth studies of thermal comfort, the heat and moisture exchange between the ambient conditions, the cockpit, and the pilot also need to be considered.

4 TLMSimulator

The TLMSimulator is an open source master simulation tool originally developed by SKF, in collaboration with Linköping University, for coupling models of bearings with models from external tools [15]. The TLMSimulator is a framework for asynchronous TLM-based co-simulation. The tool's source code has been donated to the Open Source Modelica Consortium and a graphical user interface has been developed as part of the *OpenModelica Connection Editor* [16]. The master simulator is thus available both as a stand-alone tool and as a plugin to the OpenModelica environment. The OpenModelica plugin is used in this study. Models integrated in the TLM-based framework are interconnected using TLM elements, where each element has its own independent time delay Δt_{TLM}, see Equation (1) and Equation (2). Models are therefore numerically isolated from each other, which eliminates the need for complex algorithms to decipher the necessary order of model execution. Cyclic dependencies are identified by both Cremona et al. [17] and Galtier et al. [18] as a potential problem for the non-TLM-based simulation and development environments *FIDE* and *DACCOSIM*. A cyclic dependency is the occurrence of a direct dependency between an FMU input to one or more of its outputs. The TLM method avoids this issue by design as a result of the previously mentioned inherent time independence. This time independence enables each model to have its own independent step size and solver algorithm. The framework supports the FMI standard and FMUs can be imported using a wrapper executable. The framework does, however, not fully exploit the benefits of TLM in conjunction with FMI for co-simulation. As is, a brute force method of applying a number of FMU execution

Table 2: Application example sub-models exported as FMUs. FMUs for co-simulation are denoted CS *and FMUs for model exchange are denoted* ME

	Dymola	Simulink	OpenModelica
ECS H/W	ME/CS	-	ME
ECS S/W	-	ME/CS	-
Cockpit	ME/CS	-	-
Consumer A	ME/CS	-	-
Consumer B	ME/CS	-	ME
BC	-	-	ME/CS
Engine	ME/CS	-	ME

sub-steps to cope with interpolation of input variables is implemented in the framework. FMU inputs supplied at time t_1 are valid at $t_2 = t_1 + \Delta t_{TLM}$ seconds. This means that the FMU needs to determine all data necessary between $t_1 < t < t_2$ by interpolation unless an explicit fixed step solver with step length close or equal to the global time step is implemented. In the current version of the tool, the FMI function *fmiDoStep* is called a, by the user, specified number of additional times during one global time step. The FMU can then access an interpolation table located in the master to receive linearly interpolated data. This method cannot guarantee numerical stability and it results in a limited exploitation of the advantages in computational performance associated with TLM. Even so, support is provided for both FMI for co-simulation and FMI for model exchange. For the latter, the *CVODE* and *IDA* solvers from the SUNDIALS suite are provided. A constructed simulator configuration, referred to as a composite model in the TLMSimulator, is stored in an xml file format. The xml file contains information on what FMUs are included in a particular simulator configuration, I/O connections, parameter settings, and simulation settings such as simulation start and stop times.

5 Implementation

Dymola is used as reference in terms of numerical and functional verification of the application example. The reference implementation does not include any FMUs, only Modelica models developed in Dymola and OpenModelica, see Figure 5. All of the included models comply with the current version of the Modelica standard. The models included in this reference are connected via causal connections such that each individual model can be directly exported as an FMU. The TLM connections of the TLMSimulator are represented by node components, see Figure 4, placed between each sub-system Modelica model in the reference implementation. These node components are illustrated as blue and green circles in Figure 5.

The TLMSimulator currently supports FMI 2.0 for both co-simulation and model exchange. The variable step solver *CVODE* is used when applicable (FMUs for model exchange) in the application example. The architecture of the application example TLMSimulator implementation is described in the xml format supported by the tool. The OpenModelica graphical editor was used for manipulation of the simulator architecture. The different FMUs exported for TLMSimulator integration are listed in Table 2. FMUs for both model exchange and co-simulation are exported when possible. Export of FMUs for co-simulation is only possible implementing the fixed step explicit Euler solver in the used version of OpenModelica. The robustness and simulation efficiency are increased for all sub-models listed in Table 2 if a variable step solver is implemented. Only one single sub-model is therefore exported as co-simulation from OpenModelica as a proof of concept. The tool from where each FMU is exported is stated in the top row of the table. The export from Simulink is done using the Simulink toolbox *FMI Kit for Simulink* [19] provided by Dassault Systemes. The *FMU Compliance Checker* [4], a verification tool provided by the Modelica Association, is used to ensure FMU compliance with the FMI standard.

5.1 Results

This section covers simulation results from simulations of the application example Dymola implementation. A simple mission is simulated where the altitude is linearly increased from 0m to 800m in 80s. The Mach number is increased from 0.4 to 0.9 during the same time span. The engine bleed pressure and temperature are held constant at 1MPa and 177°C, respectively. Results at the interface of each included sub-system are provided. These results will serve as reference for implementing the application example in any other FMI supporting tool. The position of the six motorized ECS valves are shown in Figure 6 for an executed reference simulation of the application example in Dymola. This implementation is done using only Modelica models. The ECS Control model is specified to start controlling the ECS after 3 seconds. The four valves controlled by the software are the valves regulating the cockpit comfort air temperature, pack outlet pressure, avionics input flow of coolant air, and pack outlet temperature. As visualized in the figure, the position of the valves controlling the cockpit input mass flow and the ejector flow is kept constant throughout the simulation. The results indicate that the developed software model prototype operates as intended. The input mass flow to the two subscribers of ECS conditioned air, Consumer A and Consumer B in Figure 1, is shown in Figure 8. Consumer A receives the desired mass flow of 30g/s after approximately 30 seconds of simulation time. This simple consumer model mimics the simplified behaviour of for example an aircraft's fuel system. Consumer B extracts a periodically varying mass flow from the ECS. Such a consumer behaviour is representative of an *On-Board Oxygen Generation System* (OBOGS) which periodically switches between different beds of porous molecular sieve type material. Figure 7 shows values of simulated cockpit pressure as well as its targeted set point. The initial value of cockpit pressure deviates significantly from the set point as a result of a high pressure initial value currently necessary during application example initialization. The pressure settling time is shown as approximately 15 seconds in the figure.

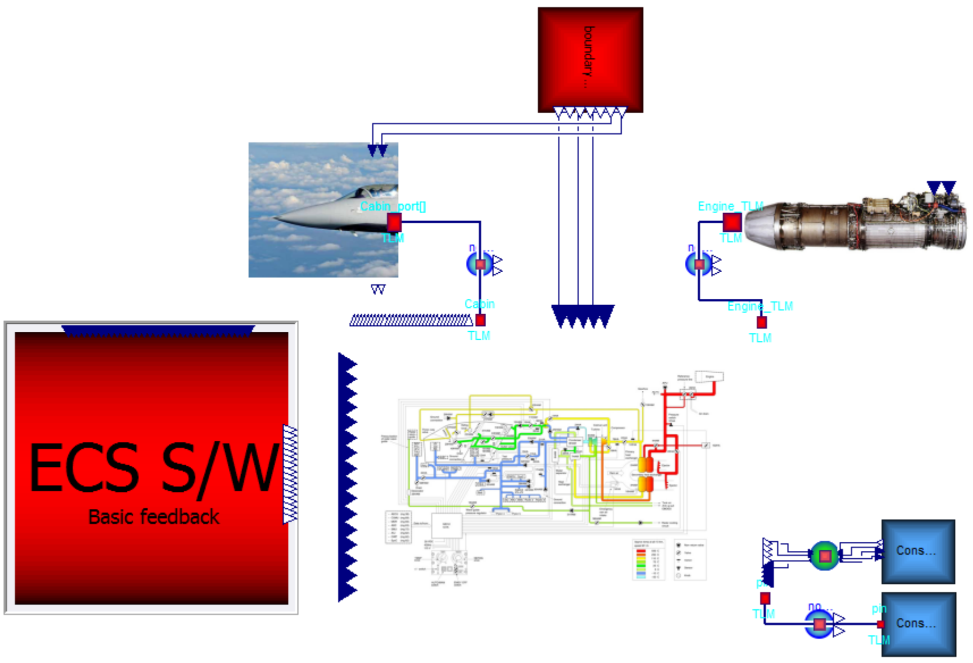

Figure 5: Graphical view of application example implementation in the commercially available modelling and simulation environment Dymola. All top level connections are causal; however, the majority of them are expressed solely in the code layer. The simulator layout is equivalent to that presented in Figure 1

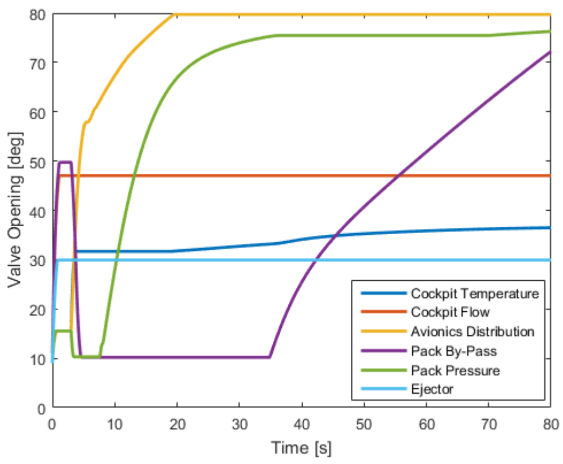

Figure 6: Position of ECS control valves during reference simulation

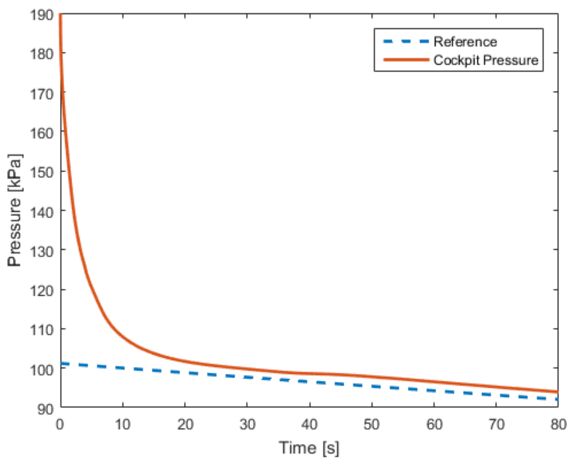

Figure 7: Simulated values of cockpit pressure along with the cockpit pressure set-point

6 Conclusions

An aircraft vehicle systems small-scale simulator is developed and implemented in Dymola. Reference simulations of this simulator are successfully conducted. The simulator can be used for evaluation, within the frame of aircraft vehicle systems simulation, of any FMI supporting master simulating tool, or any Modelica-based simulation environment. A simple and comprehensive method for separating aircraft vehicle system models, for applicability with TLM without significant additional introduced overhead, is presented along with examples from two different Modelica libraries. All separated sub-models of the application example have been exported as Functional Mock-up Units from the tool in which the sub-model was developed (Dymola, OpenModelica, and Simulink) and each individual FMU is determined as compliant with the FMI standard according to an official verification tool (the FMU Compliance Checker). The application example is implemented in the open source simulation framework supporting both FMI 2.0 and TLM; all of the application example FMUs are imported and the simulator architecture is specified using the tool's xml schema. However, no simulation results of the complete application example TLMSimulator implementation are available at the time of writing. The simulator configuration is a prototype

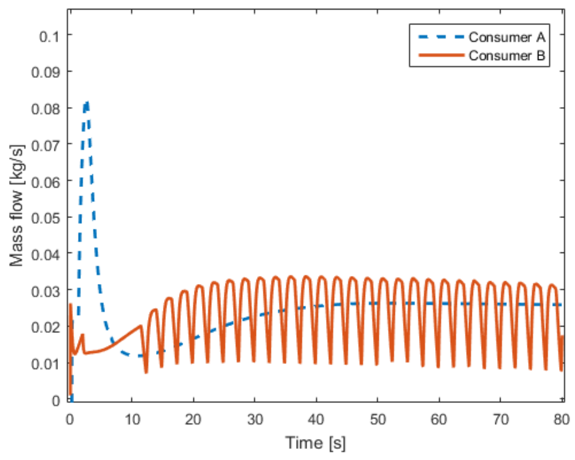

Figure 8: Simulated values of mass flow consumed by Consumer A and Consumer B

intended to be an industry-grade test case during TLMSimulator development. The application example has proven to be of great use during this development in spite of merely being a prototype and it is expected to increase in importance as it evolves.

6.1 Future work

Firstly, application example simulations in the TLMSimulator are to be conducted for verification of the application example as well as the TLMSimulator approach to simulating coupled aircraft vehicle systems. Furthermore, the developed application example is to be considered as industry-grade; however, several of the included models are severely simplified in comparison to what is being used for development etc. in the aeronautical industry. The presented small-scale simulator is an early prototype and further development is needed specifically in terms of increased detail of the sub-systems interfacing the ECS hardware. Also, industrial desktop simulators can include anything from a single sub-system to hundreds. Further inclusion of interfacing sub-systems is therefore essential. The development of the TLMSimulator is continuing within the frame of the OpenCPS project. Methods to fully exploit TLM in conjunction with FMI are currently under investigation. One approach is to utilize information on input derivatives to estimate FMU inputs in-between each global step. This development will run in parallel with the fine-tuning, and final implementation of the aircraft vehicle systems simulator application example.

7 Acknowledgements

This research has been funded via Saab Aeronautics and the ITEA 3 project Open Cyber-Physical System Model-Driven Certified Development (OpenCPS).

References

[1] International Council of Systems Engineering (INCOSE). Systems engineering vision 2020. Technical report, INCOSE, 2007.

[2] FMI development group. Functional mock-up interface for model exchange and co-simulation. Technical report, Modelica Association, 2014.

[3] Open cyber-physical system model-driven certified development. `https://www.opencps.eu/`. Accessed: 2017-03-14.

[4] Funtional mock-up interface. `http://https://www.fmi-standard.org`. Accessed: 2017-02-21.

[5] Modelica and the modelica association. `https://www.modelica.org/`. Accessed: 2017-02-21.

[6] Petter Krus. An introduction to modelling of transmission lines. Technical report, Linköping University, The Institute of Technology, 2006.

[7] Robert Braun and Petter Krus. An explicit method for decoupled distributed solvers in an equation-based modelling language. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Berlin, Germany, 2014.

[8] Henric Andersson and Magnus Carlsson. Saab aeronautics handbook for development of simulation models : Public variant. Technical Report 12/00159, Linköping University, Machine Design, 2012.

[9] Magnus Eek, Hampus Gavel, and Johan Ölvander. Definition and implementation of a method for uncertainty aggregation in component-based system simulation models. *Journal of Verification, Validation, and Uncertainty Quantification*, 2, 2017.

[10] J. Shetty, C.P. Lawson, and A.Z. Shahneh. Simulation for temperature control of a military aircraft cockpit to avoid pilot's thermal stress. *CEAS Aeronautical Journal*, 6(2):319–333, 2015.

[11] Dassault Systemes AB. *Dymola User Manual*, 1 edition, September 2016.

[12] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3*. IEEE Press, 2nd edition edition, 2015.

[13] Yutaka Hirano, Satoshi Shimada, Yoichi Teraoka, Osamu Seya, Yuji Ohsumi, Shintaroh Murakami, Tomohide Hirono, and Takayuki Sekisue. Initiatives for acausal model connection using fmi in jsae. In *Proceedings of the 11th International Modelica Conference*, 2015.

[14] Peter Fritzson and Peter Aronsson. The openmodelica modeling, simulation, and software development environment. *Simulation News Europe*, 44(45), dec 2005.

[15] Alexander Siemers, Dag Fritzson, and Iakov Nakhimovski. General meta-model based co-simulations applied to mechanical systems. *Simulation Modelling Practice and Theory*, 2009.

[16] A. Mengist, A. Asghar, A. Pop, P. Fritzon, W. Braun, A. Siemers, and D. Fritzon. An open-source composite modeling editor and simulation tool based on fmi and tlm co-simulation. In *Proceedings of the 11th International Modelica Conference*, 2015.

[17] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A. Lee. Fide - an fmi integrated development environment. In *Symposium on Applied Computing*, April 2016.

[18] Virginie Galtier, Stephane Vialle, Cherifa Dad, Jean-Philippe Tavella, Jean-Philippe Lam-Yee-Mui, and Gilles Plessis. Fmi-based distributed multi-simulation with daccosim. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, DEVS '15, San Diego, CA, USA, 2015. Society for Computer Simulation International.

[19] Dassault Systemes AB. *FMI Kit for Simulink*, October 2016.