Aerospace Technology Congress, 8-9 October 2019, Stockholm, Sweden
Swedish Society of Aeronautics and Astronautics (FTF)

Towards a Complete Co-Simulation Model Integration Including HMI Aspects

Ingo Staack[*], Jörg Schminder[*], Owais Shahid[**], and Robert Braun[*]

[*]Dept. for Management and Engineering, Linköping University, Sweden
[**]National Electric Vehicle Sweden AB (Nevs)
[*]E-mail: ingo.staack@liu.se

Abstract

Modern aircraft can be seen as heterogeneous systems, containing multiple embedded subsystems which are in today's simulations split into different domain-specific models based on different modelling methods and tools.
This paper addresses typical workflow-driven model integration problems with respect to model fidelity, accuracy in combination with the selected abstraction methods and the target system characteristics. A short overview of integration strategies with the help of co-simulation frameworks including an analysis of the inherent problems that emerge because of different domain-specific modelling methods is being given. It is shown that huge benefits can be reached with the help of a smart system break-up.
In detail, the discrepancy between the cyber-physical system simulations and human-machine interaction (HMI) models are being analysed. Therefore, a close look on typical shortcomings of behavioural models are being discussed, too.
To enable an effort-less human-in-the-loop integration into a cyber-physical system simulation, the usage of flight simulation software, offering real-time capability and a graphical user interface is suggested. This approach is applied to overcome today's complexity and shortcomings in human psychological models. An example implementation based on a commercial flight simulator software (X-Plane) together with a high-performance system simulation tool (Hopsan) via UDP communication is presented and analysed.

Keywords: flight simulator, model fidelity, co-simulation, mission simulation, workflow-driven integration, human-machine interaction, behavioural model, psychological model

1 Introduction

The demand for more efficient airplane increases steadily and as a result, the complexity of these airplane escalates as well. System simulations are extensively used to design, handle and maintain the understanding of such a complex product. It supports the designer on various tasks, to early detect possible design errors, performing design optimizations and enables for complex design analysis such as operational and maintenance concepts. First with excessive use of simulations a holistic whole product life-cycle analysis becomes possible.

Aircraft include several on-board systems such as hydraulic, pneumatic, mechanic or electric subsystems which operate in unison to fulfil a mission. It is crucial that each subsystem is performing optimal, and any possible flaws or malfunctions can be recognized and resolved early in the design process. Simulating these subsystems in a virtual test environment already during the system architecture work allows for design modifications and improvements prior to any physical testing. This enhances the design process by making it more efficient by reducing early the design uncertainties, thus enhancing the level of credibility.

2 Multi-domain Co-Simulation Frameworks

2.1 Motivation

Nowadays whole life cycle-focused product development requires a vast number of simulations to be performed at various disciplines facilitating various modelling methods that differ between the modelling task, the analysis task and the type of analysis (see [1] for an overview of different level of interest during product development work).

Any holistic model-based product development work has to include multi-domain co-simulations addressing:

- detailed sub-system simulations to study certain domain-specific system characteristics with respect to the overall system architecture
- *human-machine interaction* (HMI): the human is part of the control system or influences directly the mode of operation or use of the product.

Furthermore, it is crucial to pay attention for easy model exchange or on-the-fly model replacements in order to enable:

- **rapidly model adaption** for different analysis and design studies
- **model fidelity alteration** (preferably step-less) that fits to the available design information to enable a task/workflow-driven design process
- **model reuse** with limited adaption effort to boost development efficiency, preferable also involving black box models from third party suppliers.

2.2 Co-simulation Strategies and the Problem of Complexity

Recent research projects on simulation and modelling frameworks do focus on the above mentioned topics flexibility, adaptability and model re-usability such as the AGILE [2] or the OM-simulator [3] projects. Also, commercial tool vendors support various inter-disciplinary integration environments (such as modeFRONTIER, LMS Amesim, RCE, TechnoSoft AML, ANSYS) that allow a effort-less tool integration by supporting communication protocols, workflow process control and optimization algorithms. These environments are often denoted as *multi-disciplinary design optimization* (MDO) tools, partly enabling distributed software execution via the internet [4].

On a multi-domain cyber-physical system – and especially on a *system of systems* (SoS) – a large number of different modelling methods are composed together depending on the analysis task, the required and reasonable model accuracy and uncertainty, the available level of input information (growing over the project time), the system domains and so forth. Thereby, cyber-physical simulation models make often use of continuous time methods to model the physics, like MODELICA or different *computational fluid dynamics* (CFD) and *finite element method* (FEM) solvers. For the the control and information flow (the cyber part), discrete time simulation methods might be preferred while the behavioural model may be realized by *agent-based method*s (ABMs). Consequently, in order to compose a complete cyber-physical co-simulation, this requires to interconnected models from different domains – typically with different time constants – such that they work seamlessly together with acceptable performance and numerical robustness.

A feasible solution, shown by Fritzson et al. [5], is to use the *transmission line method* (TLM), a modelling approach which decouples sub-models using physically motivated time delays. This ensures numerical stability, and is especially suitable for real-time simulations. TLM can also greatly reduce simulation time, as was shown by Braun et al. [6] and Sjölund et al. [7]. With such strategies, making advantage of the model method specific advantages and still at the same time allowing the user to use the most suitable modelling method for the problem at hand, significant improvements to the design process, time and user-friendliness can be achieved without the need to develop new modelling tools.

Connecting inherently different simulation tools is facilitated by the *functional mock-up interface* (FMI), a tool independent standard for co-simulation and exchange of dynamic models supported by more than 100 simulation tools [8]. FMI can be used in conjunction with the the upcoming companion standard *distributed co-simulation protocol* (DCP), which facilitates communication and integration of models and real-time systems [9]. The OMSimulator master simulation tool currently supports FMI and TLM using TCP/IP connection to external tools. Possibilities of combining FMI with TLM was investigated in [10].

2.3 Model Fidelity, Complexity and Characteristic

Both terms, model complexity and model/simulation fidelity[1] are vague and no cross-domain application-independent valid standard has been establish so far (see e.g. definitions by [12, 13]) which to a large extend depends on the vast number (and weightings) of *fidelity criteria* [14–16]. Both, model complexity (in terms of the size of the overall design space) and model refinement can be expressed by the design information entropy [17].

The concept of abstraction – thus the model method and modelling fidelity – has to fit both the analysis needs *and* the systems characteristic. At a glance, any complex system model can be split-up into an structural and an behavioural part [15]. More in detail, refined taxonomies of the system's (or SoS's) characteristics like Gideons et al. [18] can be used. More in detail, the systems characteristic can be defined by the target system's properties. A selection of relevant properties to describe the characteristic type of an system is given in fig. 1.

2.4 Model Type by System Properties

Most complex systems incorporate some kind of a behavioural model part which can result in a stochastic behaviour. This behaviour can be anything from an easy control system (e.g. fuel tank filling/emptying sequence), a complex control system (e.g. an autopilot or other driver assistance systems) as well as any user-interactions on or within the system (such as pilots, flight controller, etc.). On a higher level, these parts of the system are often responsible for the way of operation of a system including more complex tasks such as operational strategies, tactics and doctrine. As a consequence, there is a unique combination of the model fidelity for each system depending on the system type (stated by the system properties shown in fig. 1). Figure 2 shows this differences in the desired degree of detail of the physical- and the behavioural model of three different systems.

In reality however, the desired direction of refinement is not reached as straight, smooth and continuous as indicated by the arrows for the three systems. Model refinement occurs instead in discrete steps, often within a single domain only by either refining an existing model or replacing a modelling technique with another abstraction method (and thereby most probably also replacing a tool within the co-simulation framework by another). A typical path of model refinement (of system b) is indicated in fig. 2 by the black stars. Using a priori a high-fidelity model techniques already from the beginning (in order

[1]Fidelity definition by [11]: "The degree to which a model or simulation represents the state and behaviour of a real world object or the perception of a real world object, feature, condition, or chosen standard in a measurable or perceivable manner; a measure of the realism of a model or simulation."

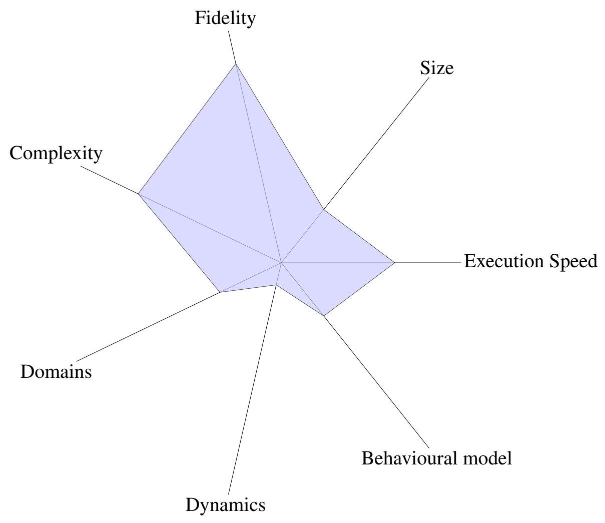

Figure 1: Radar plot of seven selected properties describing the type/characteristic of an system .

to skip the later tool transition) is often not applicable due to the absence of information to create the model and the usually lower execution speed of higher-fidelity models.

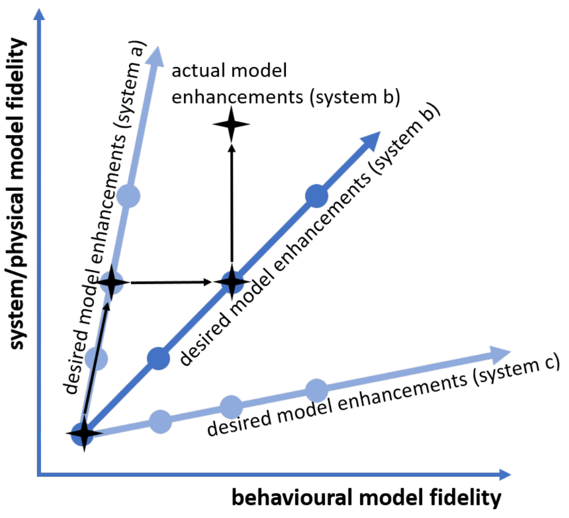
system/physical model fidelity
behavioural model fidelity
actual model enhancements (system b)
desired model enhancements (system a)
desired model enhancements (system b)
desired model enhancements (system c)

Figure 2: The ways of desired and actual model refinements within projects of different model type (characteristic).

2.5 HMI and the Need of a Real-time System

The above mentioned relationship of matching domain-specific models within a co-simulation environment means that each model domain has to be enhanced in fidelity thus reaching a certain level of certainty (or better said, reducing the level of uncertainty). The model fidelity of the physical part is rather simple to address: the model has to be able to capture the correct physics of the problem at hand. On an aircraft example, this might be already true for an object-oriented component-based (e.g. pumps, pipes, actuator, generator, etc.) approach. But what about the behavioural (control) model? The latter – often implemented in ABM – becomes already on moderate fidelities very complex, especially if simulations also include the handling of abnormal situations such as component failure simulations. Additionally with one or several *human-in-the-loop* (HITL), the behavioural model has to represent different control action levels ranging from doctrine, strategy, tactic, operations down to the respective control schemes on system element level.

3 Human-machine Interaction Modelling

Until today, the aspects of the human's physiology as well as psychology are only insufficient considered in modern simulation models. In addition, the human-machine interface of modern system must not be the real interface of the target system as indicated in fig. 3. Today's technologies often do not return the actual behaviour of a system back to the user but a human adjusted *virtual* reality. While these systems, often denoted as assistant systems, improve in many cases safety and comfort, it requires at the same time a good understanding of how the operator perceive these inputs and react on them. These is gaining on importance if modern assistant systems are applied. For example, a decoupling of the input control stick forces, as it is common in any control-by-wire setup on modern aircraft could give the pilot a false understanding of the actual aircraft state. On the opposite side, there is a necessity that the system is interpreting the actual user inputs correctly under all circumstances.

3.1 Human Psychological Modelling

A common approach used to incorporate human physiological and mental factors in simulations is by treating the human as a machine-like stimulus-response (transfer function) system although this approach does not correspond to reality at all [19]. People's individual differences such as age, sex, physical fitness, ability to take decisions, reaction time, capacity for remembering, motivation, social interactions and creativity to solve a problem, require a much more appropriate representation of the human in simulation models.

To adequately forecast the effect of human performance in a complete human-machine simulation model, dynamic and integrated computational physiological and cognitive modelling, as depicted in fig. 3, is required. Even though this is the right approach to tackle the problem, it remains challenging to develop reliable models of physiological functions or cognitive architectures [20, 21]. Problems such as validation, scaling or model simplification must be solved first before these methods can meet their full potential [22, 23].

3.2 The Need of Humans in System Simulation

Although complete human modelling is in many cases still at a basic research level, this does not mean that the human-machine interactions cannot be simulated in a sufficient manner. An alternative solution is the traditional HITL approach where actual humans interact in real-time with simulation models. A major drawback of the HITL method is the need for access to probands, which can be in particular difficult when highly trained people are needed, such as pilots. Another drawback of HITL is the exact reproducibility of simulations. While a physiological/cognitive model could provide

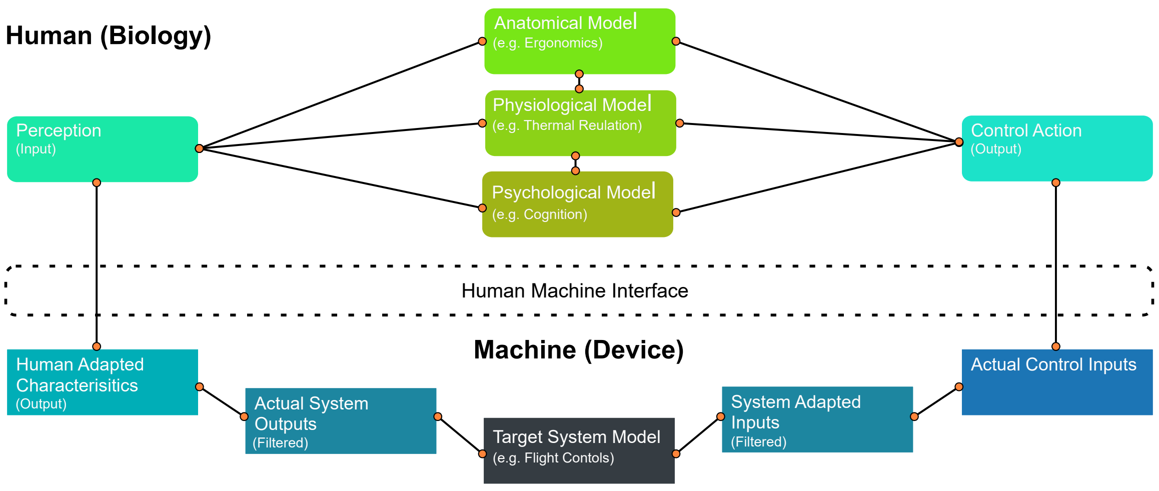

Figure 3: Human integration into the cyber-physical system context.

the same output under equal boundary conditions, is this for a HITL approach hardly to reach resulting in stochastic events. Furthermore, applying HITL, simulation speed is limited to real-time only, whereby this approach becomes inapplicable for high-fidelity co-simulations, which run often on much slower computational speed, low-fidelity simulations that run multiple times faster than real-time and any kind of optimization task that requires an enormous amount of simulation runs.

Nevertheless, no matter which of the two previous mentioned approaches is chosen, a realistic feedback from the human (model) back to the machine model can only be provided if the environment is realistic and rich in detail. A sufficient representation of the system's external and internal environmental conditions should apart from visual and aural information also include characteristic such as forces, thermal loads or odours. This high environment fidelity is necessary since the humans' reaction to this inputs are deeply context specific [14]. Integrating a flight simulator in a dynamic human-machine co-simulation can contribute to provide such a required environment.

4 Implementation Strategy & Realized Use Case Example

The following example shows a test case implementation aimed to assess a co-simulation setup of a flight simulator environment including aircraft on-board systems simulation to enable a HITL simulation and pave the way for a future human-model extension. Since the project aims to illustrate the possibilities of real-time co-simulation, the focus will be on the co-simulation integration of the example models into `X-Plane` rather than on the on-board system simulation models themself.

4.1 Related Work

A research project at Nanchang University developed a flight control model in `Simulink` for a helicopter implemented in `FlightGear` [24] including a detail description of the applied communication protocol messages setup. The result was a simple data collection and a verification process illustrating the possibility of a real-time flight simulation including external tools. A somewhat similar project was published by the Instituto Tecnológico de Aeronáutica (ITA) which aimed to simulate a quadrocopter using `MATLAB/Simulink` and the flight simulator `X-Plane` [25]. Here the (hover) control system of the quadrocopter was implemented in `Simulink`, while the vehicle's physical properties were modelled within the flight simulator environment. At Vives University College the open source platform `FlightGear` had been used in a conceptual *unmanned aerial vehicle* (UAV) design project aiming to execute electro-thermal analysis in Simulink by applying HITL simulations [26]. Also other publications (such from A. Bittar et al. [27,28]) have utilized the `X-Plane` simulation environment for *hardware/power in the loop* (H/PITL) simulations incorporating `MATLAB/Simulink` with focus on new control/actuator concepts.

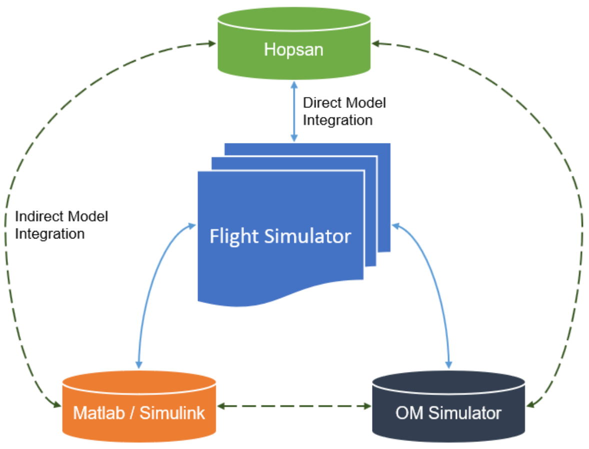

Figure 4: A simple flowchart illustrating the basic model exchange setup.

4.2 Framework Setup / Model Integration

Figure 4 shows the basic configuration of the realized co-simulation setup (exclusive the not yet realized `OMSimulator` part). The aircraft and the on-board systems are implemented in `Hopsan`, based on one example model slightly adapted to be co-simulated with `X-Plane`. It contains three main subsystems; propulsion-, control- and hydraulic actuation systems. A *six degrees of freedom* (6DoF) sub-/supersonic airplane simulation model from the standard component library is used to calculate the airplane dynamics (see component `6DOFSS` description in [29]).

On a Windows PC, `X-Plane 11` comes along with a *user datagram protocol* (UDP) communication to interact in pseudo real-time with other software. This enables co-simulations including aircraft subsystems and HITL together with `X-plane` (see fig. 5). UDP communication works with specific allocated ports from/to which messages, known as datagrams or packets, are sent or received [30]. `X-Plane 11` supports up to 138 output variables via the UDP connection. Each selection can consist of multiple variables. A complete list can be found in the `X-plane` UDP package description [31]. `Hopsan` models can be co-simulated in `Simulink` by converting them into S-functions making use of executable, pre-compiled `Hopsan` *dynamic-link librarys* (DLLs), see fig. 6. In order to establish the input and output ports of the HOPSAN model, S-function interface components must be used for the variables of interest. This allows `Hopsan` models to be integrated into `X-Plane` indirectly via `Simulink`, which supports UDP.

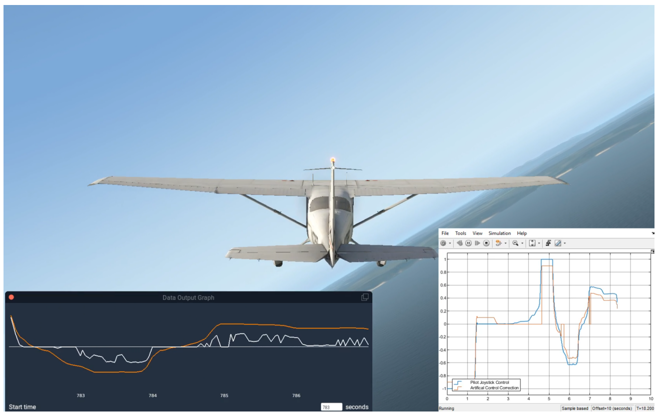

Figure 5: Screenshot of the user interface in X-Plane 11 with the X-plane datalogg to the left and the Simulink model parameter plot to the right.

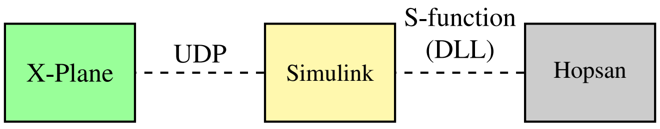

Figure 6: Communication setup of the simulation. Simulink communicates with X-plane using UDP sockets. Hopsan models are imported to Simulink as S-functions.

4.3 Model Analysis and Performance

The described UDP communication based framework has been tested in three different setups to identify the performance and bottlenecks of this integration method. To get valid results, the test have been performed on two different standard PCs running on Windows 10.

I. Pilot control test:
Time step delays and signal loss during transmission of data between the tools has been tested with a model setup of having once the pilot on the controls, recording all inputs. Afterwards, the model is fed with the recorded pilot commands again (open loop control).

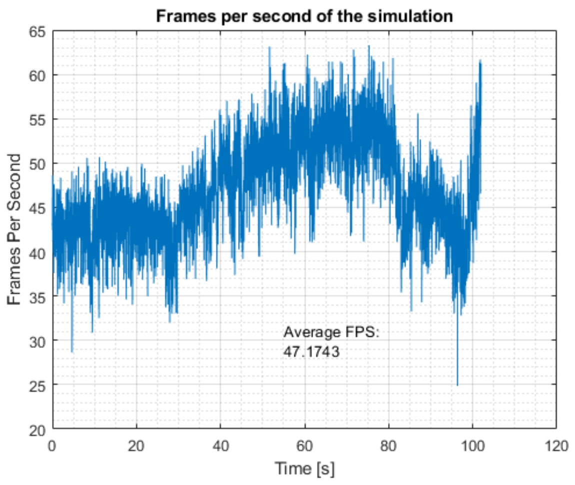

Figure 7: The frames per second model update frequency of the X-Plane model.

Figure 7 shows the performance of the `X-Plane` flight model in terms of *frames per second* (FPS). The average FPS rate during the simulation period is 47.2 with a significant fluctuation between 63 and 25 FPS. This is circa four times slower than the fixed calculation step size of 0.005 [s] or 200 [Hz] of the `Hopsan` model. The performance on the second PC was in the same range with an average FPS of 57 and even higher extrema (min: 20; max: 72).

II. Flight manoeuvrer model:
In this setup, the aircraft model including the subsystems, the flight controller and the mission controller were implemented in `Hopsan`. Consequentially, `X-plane` was only used in this setup to provide a graphical user interface, ensure (pseudo) real-time clocking. The model showed no evidence of package losses, however some flight controller parameter tuning problems arose. The reason of this model instability could not be found but may relate to the lesser update rate of the flight simulator (ca. 50[Hz]) than the simulation model time-steps (200 [Hz]). Also, round-off errors during the communication chain (due to truncations because of the different parameter normalizations in the applied tools) may have contributed to this model behaviour.

III. Pilot-in-the-loop simulation:
In this test-setup, the pilot is in the loop giving command inputs (via joystick, pedals and power lever) to the flight controller which in this setup is a simple wing leveler only. The flight controller is implemented in `Simulink` while the aircraft is modelled within `X-Plane`.

Figure 8 shows the aileron signal of the pilot control and the correction from the flight controller. It can be noticed that the flight controller signal calculated in `Simulink` corrects the aircraft to level flight by compensating with additional aileron commands. The `Simulink` model is able to compute and transmit the control signal required to restore the aircraft to level flight even with sudden, high amplitude inputs (see fig. 8 time range between 25 to 35 seconds). Due to the

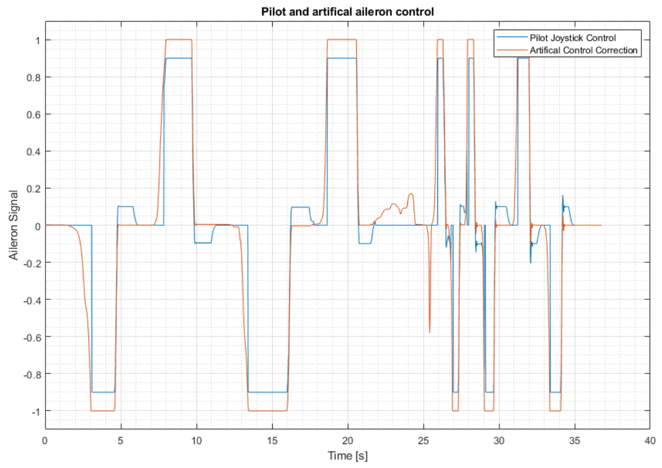

Figure 8: Aileron control signal from the pilot (red) and the Simulink roll control model (blue).

slightly higher calculation effort of the `Simulink` model in this setup, the update rate went down to 39.6 FPS in average (min: 32; max: 42.5).

4.4 Future Work

The above described implementation work is only the starting point towards a holistic scalable co-simulation model including HMI aspects. Future work (specific to this project) is going to address the following topics:

- (project specific) implement UDP communication in `Hopsan` and test the performance without `Simulink/MATLAB` in the simulation framework
- comparative studies on the existing UDP-based co-simulation framework with implementation strategies making use of a integration environment (including design process control, automation and optimization functionality) such as `RCE` [32], `OMSimulator` [33] or `modeFRONTIER` [34].
- investigate useful methods/tools to integrate H/PITL simulations into the framework, eventually making use of `dSPACE` or `LMS Amesim` (commercial tool vendors).

Figure 9 shows the planned future setup using the `OMSimulator` framework together with UDP, DLL and TCP/IP communication.

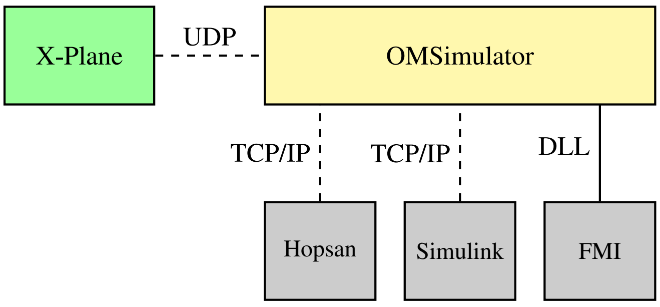

Figure 9: Possible future communication setup, using OMSimulator as the master communication unit. Other tools can be connected to OMSimulator using TCP/IP sockets, and FMI units can be imported directly.

5 Conclusion

This paper shows that it is possible to execute a flight simulation with the human (pilot) in the loop including detailed on-board system simulations down to hardware component level (such as actuators, power electronics and fuel-pumps) by co-simulation on a standard PC.

In the presented test implementation, no dedicated co-simulation environment tool has been used. Instead, a workaround using `Simulink` with its multiple interface options (here UDP and S-function/DLL) were realized. In this framework, the flight simulator works as the real-time engine and the graphical (and haptic) user interface while most simulation model calculations were performed in the co-simulated cyber-physical simulation tool.

To enable easier model handling capabilities like motivated in section 2.1, a dedicated co-simulation environment as sketched in fig. 9 should be used. This enables also other benefits such as model stability and high execution speed through its model decoupling capability as shown in section 2.2.

The typical drawbacks of a human-in-the-loop simulations has been named in section 3.2. With the limitation to real-time execution (beside other reasons), it is not suitable for many tasks during the development work. Especially optimization task that require multiple simulations cannot be executed with human-in-the-loop setups. Remaining future work to reach complete full-system simulations including simulated human behaviour is mainly needed for the psychological model, but also other aspects such as the wished continuous model refinement are not in place yet. The authors of this paper therefore suggest to work further towards universal system characteristics and model fidelity and complexity descriptions as suggested in fig. 1 and section 2.3-2.4. Such a normalization could lead to simulation guidelines that would help the product developer to choose the right simulation and simulation integration approach(es) for the problem at hand without the need of being an expert in all included model domains.

References

[1] I. Staack, K. Amadori, and C. Jouannet. A holistic engineering approach to aeronautical product development. *The Aeronautical Journal*, page 1–16, 2019.

[2] P. D. Ciampa and B. Nagel. Streamlining cross-organizational aircraft development processes: an overview of the AGILE project. In *Proceedings of the 31th ICAS Congress*, Belo Horizonte, Brazil, 2018. International Congress of the Aeronautical Sciences (ICAS).

[3] Lennart Ochel, Robert Braun, Bernhard Thiele, Adeel Asghar, Lena Rogovchenko-Buffoni, Magnus Eek, Peter Fritzson, Dag Fritzson, Sune Horkeby, Robert Hällquist, Åke Kinnander, Arunkumar Palanisamy, Adrian Pop, and Martin Sjölund. OMSimulator - integrated FMI and TLM-based co-simulation with composite model editing and SSP. In *Linköping Electronic Conference Proceedings*, pages 69–78. Linköping University, 02 2019.

[4] Petter Krus. Complete aircraft simulation for distributed system design. In *Proceedings of the International Conference on Recent Advantages in Aerospace Actuation Systems and Components*, 2001.

[5] Dag Fritzson, Robert Braun, and Jan Hartford. Composite modelling in 3-D mechanics utilizing transmission line modelling (TLM) and functional mock-up interface (FMI). *Modeling, Identification and Control*, 39(3):179–190, 2018.

[6] Robert Braun and Petter Krus. Multi-threaded distributed system simulations using the transmission line element method. *SIMULATION*, 92(10):921–930, October 2016.

[7] 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 Modeling Languages and Tools; Oslo; Norway; October 3*, number 47, pages 71–80. Linköping University Electronic Press, 2010.

[8] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for tool independent exchange of simulation models. In *8th International Modelica Conference 2011*, Como, Italy, September 2009.

[9] Modelica Association Project DCP. *DCP Specification Document, Version 1.0*. Modelica Association, Linköping, Sweden, 2019.

[10] Robert Braun, Robert Hällqvist, and Dag Fritzson. TLM-based asynchronous co-simulation with the functional mockup interface. In *IUTAM Symposium on Solver-Coupling and Co-Simulation*, Darmstadt, Germany, September 2017.

[11] DMSCO. M&S glossary - terms and definitions. [Online; accessed 2019-08-11], 2019.

[12] Manfred Roza, Jeroen Voogd, and Paul va. Fidelity considerations for civil aviation distributed simulations. In *Modeling and Simulation Technologies Conference*, number AIAA-2000-4397, 2000.

[13] Edward Burnett. A proposed model fidelity scale. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, number AIAA-2008-6689, 2008.

[14] Dennis A. Vincenzi, John A. Wise, Mustapha Mouloua, and Peter A. Hancock. *Human Factors in Simulation and Training*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2008.

[15] T. Caughlin, Donald and F. Sisti, Alex. Summary of model abstraction techniques. In *Proceedings of the SPIE 3083*, volume 3083. The International Society of Photo-Optical Instrumentation Engineers SPIE, 6 1997.

[16] T. Mowbray, Carol, C. Holter, Mark, B. Teague, Gregory, and Anddeborah Bybee. Fidelity criteria: Development, measurement, and validation. *American Journal of Evaluation*, 24:315–340, 09 2003.

[17] Petter Krus. Information entropy in the design process. In Amaresh Chakrabarti and Raghu V. Prakash, editors, *ICoRD'13*, pages 101–112, India, 2013. Springer India.

[18] James Gideon, Cihan H. Dagli, and Ann K. Miller. Taxonomy of systems-of-systems. In *Conference on Systems Engineering Research*, pages 356–363. Institute of Electrical and Electronics Engineers (IEEE), 2005.

[19] Reno Filla. *Quantifying Operability of Working Machines*. PhD thesis, Linköping UniversityLinköping University, Fluid and Mechatronic Systems, The Institute of Technology, 2011.

[20] Alessandro Vinciarelli, Anna Esposito, Elisabeth André, Francesca Bonin, Mohamed Chetouani, Jeffrey F. Cohn, Marco Cristani, Ferdinand Fuhrmann, Elmer Gilmartin, Zakia Hammal, Dirk Heylen, Rene Kaiser, Maria Koutsombogera, Alexandros Potamianos, Steve Renals, Giuseppe Riccardi, and Albert Ali Salah. Open challenges in modelling, analysis and synthesis of human behaviour in human–human and human–machine interactions. *Cognitive Computation*, 7(4):397–413, Aug 2015.

[21] M. Viceconti, D. Testi, F. Taddei, S. Martelli, G. J. Clapworthy, and S. V. S. Jan. Biomechanics modeling of the musculoskeletal apparatus: Status and key issues. *Proceedings of the IEEE*, 94(4):725–739, April 2006.

[22] Alex Kirlik. Conceptual and technical issues in extending computational cognitive modeling to aviation. In *International Conference on Human-Computer Interaction*, pages 872–881. Springer, 2007.

[23] Marc Halbrügge. Evaluating cognitive models and architectures. 2007.

[24] Jin Ying, Huazhu Luc, Jiyang Dai, and Haoman Pan. Visual flight simulation system based on Matlab/FlightGear. In *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2017*, pages 2360–2363, 2017.

[25] Helosman V. Figueiredo and Osamu Saotome. Simulation platform for quadricopter: Using Matlab/Simulink and X-plane. *Proceedings - 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium, SBR-LARS 2012*, pages 51–55, 2012.

[26] Yves C.J. Lemmens, Tuur Benoit, Rob De Roo, and Jon Verbeke. Real-time Simulation of an Integrated Electrical system of a UAV. *SAE Technical Paper Series*, 1, 2014.

[27] Adriano Bittar and Neusa Maria Franco De Oliveira. Hardware-in-the-loop simulation of an attitude control with switching actuators for SUAV. *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pages 134–142, 2013.

[28] Adriano Bittar, Helosman V. Figuereido, Poliana Avelar Guimaraes, and Alessandro Correa Mendes. Guidance software-in-the-loop simulation using x-plane and simulink for UAVs. *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pages 993–1002, 2014.

[29] HOPSAN. HOPSAN - multi-domain system simulation tool. [Online; accessed 2019-07-19], 2019. https://github.com/Hopsan/hopsan.

[30] UDP. User datagram protocol. [Online, accessed 2019-07-18], 2019. https://searchnetworking.techtarget.com/definition/UDP-User-Datagram-Protocol.

[31] X-Plane. X-Plane UDP data set output table. [Online, accessed 2019-05-22]. https://www.x-plane.com/kb/data-set-output-table/.

[32] RCE. Remote component environment (RCE): distributed, workflow-driven integration environment. [Online; accessed 2019-07-19], 2019. https://rcenvironment.de/.

[33] OMSimulator. OMSimulator - FMI-based co-simulation tool. [Online; accessed 2019-07-19], 2019. https://github.com/Hopsan/hopsan.

[34] modeFRONTIER. modeFRONTIER - multidisciplinary design optimization platform. [Online; accessed 2019-07-19], 2019. https://www.esteco.com/modefrontier.