Co-simulation: The Past, Future, and Open Challenges

Cláudio Gomes[1,4(✉)], Casper Thule[2], Julien Deantoni[5,6], Peter Gorm Larsen[2], and Hans Vangheluwe[1,3,4]

[1] University of Antwerp, Antwerp, Belgium
{claudio.gomes,hans.vangheluwe}@uantwerp.be
[2] Aarhus University, Aarhus, Denmark
{casper.thule,pgl}@eng.au.dk
[3] McGill University, Montreal, Canada
[4] Flanders Make, Lommel, Belgium
[5] University Cote d'Azur, I3S, Sophia Antipolis, France
julien.deantoni@polytech.unice.fr
[6] INRIA Kairos, Sophia Antipolis, France

Abstract. In the engineering of heterogeneous systems, there have always been challenges related to ensuring a common understanding of the interfaces between the constituent systems.

In these systems, the systematic analysis of the relevant artefacts is governed by different kinds of models based on different kinds of formalisms (e.g., state machine models for software-based controllers, and differential equations for physical sub-systems). In such a hybrid setting, it makes sense to examine how to combine different kinds of models in ways that enable a well-founded analysis of the interaction between these.

Co-simulation has been proposed as a way forward by different stakeholders in different disciplines. It is a technique to couple multiple simulation tools, so that the interactions with, and within, a coupled system can be simulated through the cooperation of these tools.

In this paper, we: provide an historical overview of the different facets of co-simulation; describe examples of industrial applications; identify the emerging trend and the challenges (both theoretical and practical) for the future use of this technology.

Keywords: Co-simulation · Complexity · System engineering

This work was executed under the framework of the COST Action IC1404 – Multi-Paradigm Modelling for Cyber-Physical Systems (MPM4CPS), and partially supported by: Flanders Make vzw, the strategic research centre for the manufacturing industry; the INTO-CPS project funded by the European Commission's Horizon 2020 programme under grant agreement number 664047; and PhD fellowship grants from the Agency for Innovation by Science and Technology in Flanders (IWT, dossier 151067).

T. Margaria and B. Steffen (Eds.): ISoLA 2018, LNCS 11246, pp. 504–520, 2018.
https://doi.org/10.1007/978-3-030-03424-5_34

1 Introduction

Integration—the interconnection of the components that comprise a system—is identified as a major source of problems in the concurrent development of complex engineered systems [62]. This is because each component is developed with assumptions and/or incomplete knowledge about other components of the system, which later turn out to be wrong [63].

To tackle these challenges, there is a need for improved development cycles, with better tools, techniques, and methodologies [65]. While modeling and simulation has been successfully applied to reduce development costs, it falls short in fostering more integrated development processes [7]. To see why, note that a model of the complete system is required for simulation, and consider the following obstacles:

- Accurately simulating a complete system model might be difficult. For example, the transient simulation of digital circuits is difficult because there are sub-circuits whose dynamics change significantly faster than others [49], forcing the simulation to be run at a prohibitively high level of detail.
- Heterogeneous systems are best modelled with a mix of formalisms [66] or example, consider a power window system [56], present in the majority of the vehicles produced today. It includes both software elements (best modelled with a Statechart like formalism), and physical elements (best modelled with differential equations based formalism).
- Subsystem models might be costly. In systems that encompass subsystems produced by external suppliers, the licensing costs required to get access to models might be too high, due to the Intellectual Property. For example, consider the exhaust gas recirculation water handling system, reported in [55], where the dirty water is pumped to a water treatment center (externally developed) to be purified and reused. As claimed by the authors, having higher fidelity models of each of the subsystems would allow the engineers to design better control strategies.
- Models of subsystems might be black boxes. At later stages in the development process, prototypes for subsystems may be coupled to models of the remaining subsystems, to enable global validation of the system. For example, the validation of the power window controller might be done by simulating the controller in a computer, and connecting it to a real motorized window [18], which is considered a black box from the point of view of the controller. Other black boxes include inductive models of subsystems, produced from extensive physical experimentation. For example, an anti-lock braking system controller might be validated against black box wear and tear models of the braking pads, to evaluate its performance when the effectiveness of these subsystems decreases [21].

A prospective concept to address the above challenges, and unleash the full potential of simulation, is collaborative simulation, also known as co-simulation [40]. This concept concerns coupling of models created in different formalisms and makes it possible to simulate the entire system by simulating its constituents

and exchanging data between them. Thus, the behavior of a coupled system is computed by the communication of multiple simulation tools, each responsible for computing the behavior of a constituent subsystem [30,44,51]. Each simulator is broadly defined as a *black box* capable of exhibiting behaviour, consuming inputs and producing outputs. Examples of simulators include dynamical systems being integrated by numerical solvers [12], software and its execution platform [16], dedicated real-time hardware simulators (e.g., [34]), physical test stands (e.g., [69, Fig. 3]), or human operators (e.g., [13, Fig. 24], [53, Fig. 6]).

Co-simulation foments a more integrated development process by allowing different teams to observe how their subsystem behaves when coupled to the rest of the system (full system analysis), while reusing the work made by the other teams. Furthermore, it improves the relationship between external suppliers and system integrators, where the system integrators can use virtual surrogates of the subsystems produced by the suppliers, to test their adequacy. With the appropriate Intellectual Property protections, these virtual surrogates can even be provided by the supplier, for increased validity.

In order to run a co-simulation, all that is required is that the participating simulation tools expose the outputs and consume the inputs, of the allocated subsystem over simulated time. The same loose requirements that make co-simulation great to integrate many different simulation tools, also raise difficult challenges.

In the following sections, we explore those challenges by first providing an historical overview of co-simulation, then examples of industrial case studies, and finally the emerging trend.

2 The Facets of Co-simulation: Historical Overview

Co-simulation is not a new concept. Instead, it is the aggregation of multiple research trends that were sparked by the advances in computer simulation techniques, and the increased demands on this field. In the following paragraphs, we summarize some of the main milestones that lead to the facets of co-simulation. Figure 1 situates these in time.

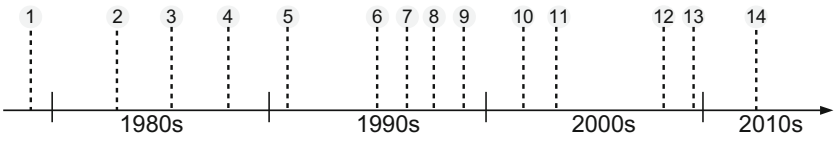

Fig. 1. Timeline of co-simulation milestones. From 1970s up to 2015.

2.1 Late 70s and 80s

(1) To the best of our knowledge, the first discrete event synchronization algorithms were published in the late seventies [39], around the same time that Lamport [42] published his seminal paper regarding the ordering of events in distributed process networks. Discrete event simulators compute the behavior of a system by isolating the most important events and computing the state evolution of the system from one event to the next [23]. The state evolution evolves discontinuously, with each discontinuity being caused by an event. In this paradigm, a coupled system can be broken down into subsystems that exchange events, which then are simulated in parallel, each in a separate process. Since processes run in parallel, and react to incoming events by updating their state, and potentially sending events, it is important to ensure the correct *synchronization* of the subsystems, so that no event happening at time t_i is processed by a subsystem which is at time $t > t_i$.

Around the same time, in the continuous simulation domain, new challenges were being uncovered. The main difference between the continuous and discrete event simulation domain lies in the fact that the state of a continuous system evolves continuously over time. Simulators of continuous systems that run in digital computers cannot compute every point of its state. Instead, they rely in the smoothness of these systems (coming from physical laws), to approximate the state evolution at countable points in time [12]. The fundamental tradeoff is: the closer one wants the time points to be, the more accurate the approximation is, but the higher the performance cost.

(2) In the late seventies and early eighties, as electrical circuits increased in size, their simulation algorithms were becoming a bottleneck in the development process because of the long simulation times. Practitioners noticed that, for sufficiently large circuits, only a small fraction of the subsystems had actively changing voltage levels, at any point in time. This led to the development of simulation techniques that, in a similar way to their discrete event based counterparts, only computed a new state of each subsystem when its outputs had changed significantly [49]. Additionally, to exploit parallelism and reduce numerical instabilities, the waveform relaxation techniques were introduced. In these, during a computation interval $t \rightarrow t + H$, each subsystem was assigned to a simulator which approximated its solution in that interval, using whatever simulation step size was required to keep the approximation error of that subsystem within tolerance. Then the simulators exchanged the solution trajectories, and were asked to re-compute the same interval, using the updated input trajectories.

These techniques made possible the simulation of large scale circuits because they exploited parallel computers, and naturally supported subsystems with different dynamics: systems which changed slowly where more quickly driven to convergence, and with larger simulation step sizes. Additionally, these techniques were subject to extensive numerical analysis [47], highlighting their interesting theoretical properties.

(3) In the late eighties, the release of the Time Warp Operating System represented the optimistic facet in parallel discrete event simulation. It acknowledged that the performance of a parallel discrete event simulation could be increased by allowing the different processes to simulate as fast as they could, and correcting causality violations. The corrections are made by rolling back the processes to a state that is consistent with the time of the event that caused the violation.

(4) The performance of optimistic discrete event synchronization algorithms was such that it sparked the research into large scale simulations with humans interacting in realistic environments created by collaborating simulators. Developed during the 80s, SIMNET was dedicated to military trainings involving thousands of simulators representing, for instances tanks or helicopters [48]. It encompasses an architecture and protocol to implement the optimistic synchronization of simulators in a distributed environment, with real-time constraints. In order to keep a reasonable level of accuracy and realism, one of the innovations is the concept of dead-reckoning models. A dead-reckoning model is a computationally lightweight version of some other model, whose purpose is to be used by interested simulators when there is a failure of communication, or when the synchronization times are far apart.

2.2 90s

(5) In the early nineties, coordination languages emerged (*e.g.*, Linda [6], Manifold [4]). These focused on the specification of the interaction between different parts of the system. According to [25], "*Coordination is the process of building programs by gluing together active pieces*". A system designer defines one or more coordination model(s) to specify how the system models interact with each other.

During the same period, the *software architecture* research field proposed languages to abstract, structure, and reason about complex systems. One example is the Architecture Description Languages (ADL) [24]. An ADL description usually specifies a system in terms of components and interactions among those components. Such languages helped (1) to clarify structural and semantics difference between components and interactions, (2) to reuse and compose architectural elements, (3) to identify/enforce commonly used patterns (*e.g.*, architectural styles).

Coordination languages and ADLs have common objectives [52]. They build/understand/analyse a system based on "components" possibly written in different languages and connectors (which include the specification of the interaction/coordination).

In 1990, United Airlines ordered 34 Boeing 777s, the first aircraft to be developed with concurrent engineering [37,38]. The design was communicated fully in digital form, later aptly named a DMU (Digital Mockup Unit [3]), using CAD

tools to showcase the different views of the system. This central repository of information served many purposes: (i) every team could consult the specifications of the subsystems made by any other team; (ii) simulations could be carried out periodically, to detect problems in the design; (iii) both the assembly and maintenance phases of the system could affect the design phase, by running simulations of repairs and assembly.

This milestone represented an increase in the information that is taken into account for the design of the product. It now did not come only from requirements, but also from other stages of the life-cycle of the system: manufacturing, assembly and maintenance. The milestone also highlights the many different purposes for which models of systems have to be available, and new kinds of simulations.

(6) As digital circuits became more complex, they comprised microprocessors running software. This field spawned the need for hardware/software co-simulation [57], highlighting the heterogeneity facet. Before using co-simulation, software developers had to develope their code with little information about the underlying hardware, leading to painful integration efforts later on. Thanks to the coupling of circuit emulators and the software execution, they were able to quickly identify miscommunication errors before building hardware prototypes.

In the field of physical system simulation, researchers realized that there should be a standardized way of representing physical system models, so that there could be easily coupled to form complex systems [50]. This was called the DSBlock (Dynamical System Block) standard [50]. This proposal later inspired a widely adopted standard for co-simulation: the Functional Mockup Interface standard. (7) While the composition of DSBlocks still needed a solver, and is therefore not strictly considered co-simulation, this was a milestone in highlighting the need for standardization for continuous system co-simulation, which was also identified as a research priority [67]. (8) SIMNET evolved into the DIS (Distributed Interactive Simulation) standard [35], for discrete event based co-simulations.

As embedded systems were enhanced with communication capabilities, researchers noticed that the simulation of these distributed systems should not always be run at the same level of detail. Instead, the designers should be able to choose the level of detail they wanted for each embedded system: from the highest level of detail (circuit simulation), to the lowest (software simulation). This highlights the facet of multi-abstraction co-simulation, and identified the main issues in coupling simulators that were in different levels of abstraction.

2.3 2000s

(9) The early 2000s was marked by multiple reported applications of co-simulation being used in industrial case studies [5,43]. These had in common one facet: two simulators were coupled, each specialized in one domain, in a feedback loop. (10) For example, in [5] the authors reports on the study of the interaction between the pantograph (a mechanical structure on top of a train, connecting it to the electric grid), and a catenary (over hanging cable that transmits electricity to the train). A flexible body simulator was used to compute the behavior of the catenary, and a multi-body simulator was used for the pantograph. (12) In the meantime, the DIS standard, and its protocols, were generalized to non-real time applications, in what became the HLA (High Level Architecture) standard [1].

(11) In order to ensure the correctness of coordinated heterogeneous model simulations, the Ptolemy and the Modhel'x projects proposed to expose some information about the behavioral semantics of languages (named Model of Computation) [9,20]. Then, they defined adaptations so that they could be co-simulated.

(13) In 2008, the MODELISAR project published the FMI (Functional Mockup Interface) standard [7], whose essential contribution to co-simulation was the concept of Intellectual Property protection. It was an evolution of the DSBlock proposal, but recognizing that each subsystem might need its own simulator. This standard is widely adopted in industry[1] [58], where the simulation of externally supplied components can be costly due to high licensing costs.

Although there was some research about the coordination of black-box physical system simulators before the FMI Standard was published (*e.g.*, [5,31,41], and other references in [29]), it does not standardize the synchronization protocol between simulators. The main reason is that, as in continuous system simulation, there is no one-fits-all simulation algorithm. This is in contrast to discrete event simulation, where the implementations of the DIS and HLA standards provide everything to run the co-simulation.

2.4 2010s

The current decade is marked by several applications of co-simulation across many domains (see, *e.g.*, [29,59]), the Digital Twin [26] concept, and an effort to systematically study co-simulation, with the publication of surveys [30,32].

(14) The Digital Twin extends the DMU concept not just to the design and assembly phases of the system, but also to the maintenance. The essential idea is to use high fidelity models of the system, calibrated from sensory information collected during its operation, to affect how the system should operate, predict failures, schedule maintenance, etc.

[1] http://fmi-standard.org/.

3 Applications

3.1 Exhaust Gas Recirculation (MAN Diesel and Turbo)

MAN Diesel & Turbo (MDT) is one of the largest producers of two-stroke combustion engines with distributed embedded control system. Due to new emissions legislation on NO_x, the systems that reduce the emission of this gas need to be improved. Since the development is split between different departments, using different tools, with limited sharing of models, co-simulation was applied to maximize reuse of models [55].

The work in [55] describes an exhaust gas recirculation system, and a water handling system. The purpose is to clean and recirculate exhaust gas to a ship engine intake manifold. The exhaust gas is cleaned by spraying water into it, and allowing the mixture to cool down and flow into a receiving tank. Then, the (dirty) water is pumped to a water treatment center (externally developed) to be purified and reused.

The initial approach consisted of developing the control system in an in-house application framework, that simulated both the control system and the physical models of the ship engine. While the traditional setup allows for simulation, the physical models are often implemented at a lower level of detail than e.g. Matlab/Simulink® models. The co-simulation approach, based on the FMI standard, coupled the in-house application to MATLAB, so that higher fidelity physical models could be used. They believe that, had this approach been used from the start, then a water tank overflow problem could have been discovered before running the software on an expensive engine test bench.

3.2 Driverless Lawn Mower (Agro Intelligence)

Another application of co-simulation is the development of a steering controller of an industrial size driverless lawn mower [22]. Besides aiding in the development of the control and navigation system of the lawn mover, co-simulation was applied to investigate alternative designs that would otherwise be both costly and time-consuming to test with physical prototypes.

The co-simulation scenario consisted of three parts: a simulator representing the vehicle dynamics, a simulator representing the control algorithm and a simulator to convert values between the two. Additionally, each alternative design was projected in a 3D animation based on the game engine Unity, that it could be visually inspected by designers and clients.

To make sure the co-simulation results were valid and accurate, an initial prototype was conceived and tested. Afterwards, multiple designs were evaluated with co-simulation, to find the optimal look-ahead distance and velocity. The simulation results for multiple look-ahead distances, and fixed velocity, are shown in Fig. 2.

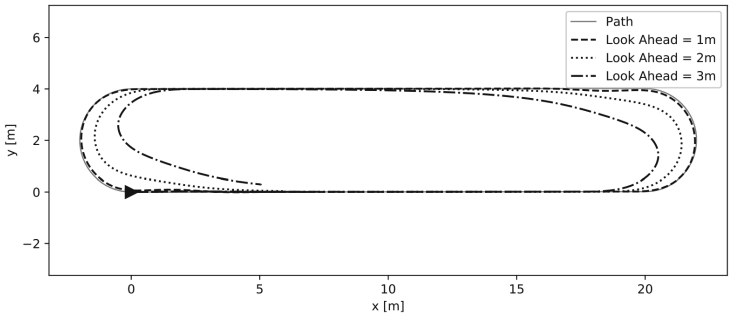

Fig. 2. Simulated trajectories for look-ahead distance with velocity 1 m/s [54]

4 Emerging Trend and Challenges

4.1 Towards Full Virtualization

Throughout the history of co-simulation, a common trend emerges: a gradual shift towards the virtualization of not just the design of the system, but also assembly, operation, and maintenance.

The virtualization of the design of the system has been one of the primary uses of co-simulation, backup by concurrent engineering processes.

The virtualization of the assembly reflects an increased demand in the information that should be taken into account at the design phase, with concepts like the Digital Mockup Unit.

Complex systems that need interaction with human operators require training interfaces. Marked by military training simulators, the virtualization of operation refers to the creation of complex training environments at almost no cost by leveraging the same co-simulation scenarios used in the design phase. As an example towards this future, we highlight the design of a motion compensated crane [14], by ControlLab, where the crane operators are trained using a virtual reality environment (see Fig. 3).

Fig. 3. 3D real-time simulation of a motion compensated crane. Taken from [14].

Finally, extending the lifespan of systems, and reducing their downtime through the virtualization of their maintenance, is becoming a priority. This means that co-simulation can be combined with advanced sensors to create smart monitors (Digital Twins) that predict failures.

4.2 Challenges

The historical overview, and main trend identified, can be used to highlight some of the challenges that researchers and industry will need to overcome in the upcoming years.

We divide these challenges into four categories: Design Space Exploration (DSE), X-in-the-Loop Co-simulation, Incremental Testing/Certification (IT/C), and Education.

Design Space Exploration consists of the systematic analysis and evaluation of different designs over a parameter space. When the evaluation involves running a co-simulation, then ensuring that co-simulations can be run quickly, accurately, and respecting the physical constraints of the system.

Since the results of these simulations are typically not inspected by experts, it is crucial that these can be trusted. To this end, we highlight the need to ensure that each configuration of the system is valid, and the need for the co-simulation to preserve any properties that the configuration of the system satisfies.

Validity refers to whether the composition of subsystem models (induced by the co-simulation scenario) reflects a physically meaningful coupled system [17,70]. This property is important because physical system models have many implicit assumptions, and their combination may violate those assumptions, purging their predictive value. For example, in [60] the authors ran a questionnaire through several experts in various domains of physics, asking them to identify the implicit assumptions in a simple model of a particle moving in a viscous medium. No expert was able to identify all the 29 assumptions, identified by their combined expertise.

The evolution of many engineered systems can be summarized by their evolution from one equilibrium to another [19], and it is important that their corresponding co-simulations reflect this property. While analyses have been developed that enable the automated verification of this property for continuous co-simulations (see [30, Sect. 4.3] and references thereof), there are many open challenges with the co-simulation of hybrid systems [27], and adaptive co-simulations [28].

X-in-The-Loop refers to co-simulations that are restricted in time and computing resources, due to the presence of human operators, animation requirements, or physical subsystems. In this context, there is a need for simulators which can provide contracts with timing guarantees on their computation time, based on the inputs and parameterization.

IT/C consists of the co-simulation activities that are applied as part of concurrent engineering activities, where the models of each subsystem are refined over time and integrated frequently. We highlight the need for co-simulations that provide formal guarantees on the accuracy of the behavior that is computed. Since the definition of correct co-simulation is elusive and depends on the domain of application, each simulator should provide some form of contract. It should be possible to obtain an abstraction of each simulation units that is appropriate to the kind of contracts defined. Existing research could be used as a starting point [8,11,36,46].

Once each simulator provides formal guarantees, then the orchestration algorithm should ensure that the composition of those contracts, and other formal properties, can be satisfied. As highlighted by works on heterogeneous simulations and more recently in [45], the way to orchestrate the different simulators can lead to incorrect results. This is especially true when discrete models (with frequent and natural discontinuities) are in the loop since a minor change in timings can result in different behavior (let consider for instance a double click versus two consecutive clicks).

To illustrate, consider a simulator that guarantees that there are no more than one discontinuity every 10 s. Then, depending on similar contracts satisfied by other simulators, a similar kind of contract could be satisfied by the co-simulation.

Education refers to those challenges that are of non-technical nature, but are nonetheless crucial to attain the full virtualization vision.

In order for companies to adopt co-simulation there are several concerns that hinder the theoretical possibilities from being employed in practical setting. One of these is the protection of intellectual property, which limits the information that is available for a given simulation unit. It is not an issue in itself, but it is an issue when considering other desirable properties of co-simulation, e.g. performance. For example, [61] describes two master algorithms, one that allows parallel computation but is limited in its applicability, and another that is less limited in applicability but requires a sequential execution. However, the information required to choose the optimal master algorithm in this case is not available. Similarly, [64] concerns precompiling a master algorithm optimised for a given scenario, but this also requires information, that is not available in a black box implementation.

Another challenge is related to the current co-simulation standards. This is described in [10], which puts forth several requirements for hybrid co-simulation, such as superdense time, and relates them to the FMI standard. In general, time representation is a very important aspect of co-simulation, and [15] presents several extensions to FMI. One of these is that in theory several theorems uses real numbers, which has infinite precision. However, these are often represented as numbers with finite precision.

Finally, proper integration with existing development processes. Co-simulations are initiated by different users with different backgrounds. This is not just about pushing a button and getting results: there is a need to integrate robust co-simulation frameworks into existing tools, such that each different kind of user can use the most comfortable tool as a front end to run the co-simulations, and that user understand what he is doing. To this end, education and technology transfer are crucial steps.

5 Conclusion

Co-Simulation holds the promise to unleash the full potential of simulation. However, it is not a new concept. In this paper we present the historical events that resulted in what is today known as co-simulation. These highlight a trend towards the virtualization of every interaction with complex systems. Based on this trend, we identify several exciting challenges that lie ahead.

References

1. IEEE. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification. IEEE Standard 1516-2010 (2010). https://standards.ieee.org/findstds/standard/1516-2010.html
2. Åström, K.J., Elmqvist, H., Mattsson, S.E.: Evolution of continuous-time modeling and simulation. In: ESM, pp. 9–18 (1998)

3. Andert Jr., E.P., Morgan, D.: Collaborative virtual prototyping and test. Naval Eng. J. **110**(6), 17–23 (1998). http://www.ingentaconnect.com/content/asne/nej/1998/00000110/00000006/art00007
4. Arbab, F., Herman, I., Spilling, P.: An overview of manifold and its implementation. Concurrency Pract. Exper. **5**(1), 23–70 (1993). https://doi.org/10.1002/cpe.4330050103
5. Arnold, M., Günther, M.: Preconditioned dynamic iteration for coupled differential-algebraic systems. BIT Numer. Math. **41**(1), 1–25 (2001)
6. Bjornson, R., Carriero, N., Gelernter, D., Mattson, T., Kaminsky, D., Sherman, A.: Experience with linda. Yale University Computer Science Department, Technical report RR-866 (1991)
7. Blochwitz, T., et al.: The functional mockup interface for tool independent exchange of simulation models. In: 8th International Modelica Conference, pp. 105–114. Linköping University Electronic Press, Linköpings universitet, Dresden, Germany, June 2011
8. Bouissou, O., Chapoutot, A., Djoudi, A.: Enclosing temporal evolution of dynamical systems using numerical methods. In: Brat, G., Rungta, N., Venet, A. (eds.) NFM 2013. LNCS, vol. 7871, pp. 108–123. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38088-4_8
9. Boulanger, F., Hardebolle, C.: Simulation of multi-formalism models with ModHel'X. In: Proceedings of ICST 2008, pp. 318–327. IEEE Computer Society (2008)
10. Broman, D., Greenberg, L., Lee, E.A., Masin, M., Tripakis, S., Wetter, M.: Requirements for Hybrid Cosimulation. Technical report (2014)
11. Carter, R., Navarro-López, E.M.: Dynamically-driven timed automaton abstractions for proving liveness of continuous systems. In: Jurdziński, M., Ničković, D. (eds.) FORMATS 2012. LNCS, vol. 7595, pp. 59–74. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33365-1_6
12. Cellier, F.E., Kofman, E.: Continuous System Simulation. Springer, New York (2006). https://doi.org/10.1007/0-387-30260-3
13. Chen, B.C., Peng, H.: Differential-braking-based rollover prevention for sport utility vehicles with human-in-the-loop evaluations. Vehicle Syst. Dyn. **36**(4–5), 359–389 (2001)
14. Controllab Products: Design of a Compensated Motion Crane using INTO-CPS. Technical report, Press Release EU, Enschede, Netherlands (2018)
15. Cremona, F., Lohstroh, M., Broman, D., Lee, E.A., Masin, M., Tripakis, S.: Hybrid co-simulation: it's about time. Softw. Syst. Model. (2017)
16. Denil, J., De Meulenaere, P., Demeyer, S., Vangheluwe, H.: DEVS for AUTOSAR-based system deployment modeling and simulation. Simulation **93**(6), 489–513 (2017). http://journals.sagepub.com/doi/10.1177/0037549716684552
17. Denil, J., Klikovits, S., Mosterman, P.J., Vallecillo, A., Vangheluwe, H.: The experiment model and validity frame in M&S. In: Proceedings of the Symposium on Theory of Modeling and Simulation, vol. 49 (2017)
18. Denil, J., Meyers, B., De Meulenaere, P., Vangheluwe, H.: Explicit semantic adaptation of hybrid formalisms for FMI co-simulation. In: Barros, F., Wang, M.H., Prähofer, H., Hu, X. (eds.) Symposium on Theory of Modeling and Simulation: DEVS Integrative M&S Symposium, pp. 99–106. Society for Computer Simulation International San Diego, CA, USA, Alexandria, Virginia, April 2015
19. Distefano, J.: Feedback and Control Systems (2013)
20. Eker, J., et al.: Taming heterogeneity - the Ptolemy approach. Proc. IEEE **91**(1), 127–144 (2003)

21. El-Garhy, A.M., El-Sheikh, G.A., El-Saify, M.H.: Fuzzy life-extending control of anti-lock braking system. Ain Shams Eng. J. **4**(4), 735–751 (2013). https://doi.org/10.1016/j.asej.2012.12.003
22. Foldager, F., Larsen, P.G., Green, O.: Development of a driverless lawn mower using co-simulation. In: 1st Workshop on Formal Co-Simulation of Cyber-Physical Systems, Trento, Italy, September 2017
23. Fujimoto, R.M.: Parallel discrete event simulation. Commun. ACM **33**(10), 30–53 (1990)
24. Garlan, D., Shaw, M.: An introduction to software architecture. Technical report, Pittsburgh, PA, USA (1994)
25. Gelernter, D., Carriero, N.: Coordination languages and their significance. Commun. ACM **35**(2), 96 (1992). https://doi.org/10.1145/129630.376083
26. Glaessgen, E., Stargel, D.: The digital twin paradigm for future NASA and U.S. air force vehicles. In: Structures, Structural Dynamics, and Materials Conference: Special Session on the Digital Twin, pp. 1–14. American Institute of Aeronautics and Astronautics, Reston, Virigina, April 2012. https://doi.org/10.2514/6.2012-1818
27. Gomes, C., Karalis, P., Navarro-López, E.M., Vangheluwe, H.: Approximated stability analysis of bi-modal hybrid co-simulation scenarios. In: Cerone, A., Roveri, M. (eds.) SEFM 2017. LNCS, vol. 10729, pp. 345–360. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74781-1_24
28. Gomes, C., Legat, B., Jungers, R.M., Vangheluwe, H.: Stable adaptive co-simulation: a switched systems approach. In: IUTAM Symposium on Co-Simulation and Solver Coupling, Darmstadt, Germany (2017). To appear
29. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: State of the art. Technical report, February 2017. http://arxiv.org/abs/1702.00686
30. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a survey. ACM Comput. Surv. **51**(3) (2018). Article 49
31. Gu, B., Asada, H.H.: Co-simulation of algebraically coupled dynamic subsystems. In: American Control Conference, vol. 3, pp. 2273–2278. IEEE, Arlington (2001)
32. Hafner, I., Popper, N.: On the terminology and structuring of co-simulation methods. In: Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, pp. 67–76. ACM Press, New York (2017). http://dl.acm.org/citation.cfm?doid=3158191.3158203
33. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems (1996)
34. Himmler, A.: Hardware-in-the-loop technology enabling flexible testing processes. In: 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, pp. 1–8. American Institute of Aeronautics and Astronautics, Grapevine (Dallas/Ft. Worth Region), Texas, January 2013. https://doi.org/10.2514/6.2013-816
35. IEEE: IEEE Standard for Distributed Interactive Simulation-Application Protocols (2012). Publication Title: IEEE Std 1278.1-2012 (Revision of IEEE Std 1278.1-1995)
36. Immler, F.: Formally verified computation of enclosures of solutions of ordinary differential equations. In: Badger, J.M., Rozier, K.Y. (eds.) NFM 2014. LNCS, vol. 8430, pp. 113–127. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06200-6_9
37. Jo, H.H., Parsaei, H.R., Sullivan, W.G.: Principles of concurrent engineering. In: Parsaei, H.R., Sullivan, W.G. (eds) Concurrent Engineering, pp. 3–23. Springer, Boston (1993). https://doi.org/10.1007/978-1-4615-3062-6_1

38. Jørgensen, N.: The Boeing 777: development life cycle follows artifact. In: World Conference on Integrated Design and Process Technology (IDPT), pp. 25–30. Citeseer (2006)
39. Kent Peacock, J., Wong, J., Manning, E.G.: Distributed simulation using a network of processors. Comput. Netw. (1976) **3**(1), 44–56 (1979). http://linkinghub.elsevier.com/retrieve/pii/0376507579900539
40. Kübler, R., Schiehlen, W.: Modular simulation in multibody system dynamics. Multibody Syst. Dyn. **4**(2–3), 107–127 (2000)
41. Kübler, R., Schiehlen, W.: Two methods of simulator coupling. Math. Comput. Model. Dyn. Syst. **6**(2), 93–113 (2000)
42. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
43. Le Marrec, P., Valderrama, C.A., Hessel, F., Jerraya, A.A., Attia, M., Cayrol, O.: Hardware, software and mechanical cosimulation for automotive applications. In: 9th International Workshop on Rapid System Prototyping, pp. 202–206 (1998)
44. Li, W., Zhang, X., Li, H.: Co-simulation platforms for co-design of networked control systems: an overview. Control Eng. Pract. **23**, 44–56 (2014)
45. Liboni, G., Deantoni, J., Portaluri, A., Quaglia, D., De Simone, R.: Beyond time-triggered co-simulation of cyber-physical systems for performance and accuracy improvements. In: 10th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, Manchester, United Kingdom, January 2018. https://hal.inria.fr/hal-01675396
46. Maler, O., Batt, G.: Approximating continuous systems by timed automata. In: Fisher, J. (ed.) FMSB 2008. LNCS, vol. 5054, pp. 77–89. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68413-8_6
47. McCalla, W.J.: Fundamentals of Computer-Aided Circuit Simulation, vol. 37. Springer, New York (1987). https://doi.org/10.1007/978-1-4613-2011-1
48. Miller, D., Thorpe, J.: SIMNET: the advent of simulator networking. Proc. IEEE **83**(8), 1114–1123 (1995). http://ieeexplore.ieee.org/document/400452/
49. Newton, A.R., Sangiovanni-Vincentelli, A.L.: Relaxation-based electrical simulation. SIAM J. Sci. Stat. Comput. **4**(3), 485–524 (1983)
50. Otter, M., Elmqvist, H.: The DSblock model interface for exchanging model components. In: Proceedings of the Eurosim 1995, Simulation Congress, pp. 505–510 (1995)
51. Palensky, P., Van Der Meer, A.A., Lopez, C.D., Joseph, A., Pan, K.: Cosimulation of intelligent power systems: fundamentals, software architecture, numerics, and coupling. IEEE Indus. Electr. Mag. **11**(1), 34–50 (2017)
52. Papadopoulos, G.A., Arbab, F.: Coordination models and languages. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands (1998)
53. Pedersen, N., Bojsen, T., Madsen, J.: Co-simulation of cyber physical systems with HMI for human in the loop investigations. In: Symposium on Theory of Modeling and Simulation, Society for Computer Simulation International, Virginia Beach, TMS/DEVS 2017, Virginia, USA, pp. 1:1–1:12 (2017). http://dl.acm.org/citation.cfm?id=3108905.3108906
54. Pedersen, N., Lausdahl, K., Sanchez, E.V., Thule, C., Larsen, P.G., Madsen, J.: Distributed co-simulation of embedded control software using INTO-CPS. In: International Conference on Simulation and Modeling Methodologies, Technologies and Applications, Madrid, Spain, July 2017. To appear

55. Pedersen, N., Lausdahl, K., Vidal Sanchez, E., Larsen, P.G., Madsen, J.: Distributed co-simulation of embedded control software with exhaust gas recirculation water handling system using INTO-CPS. In: 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 73–82. SCITEPRESS - Science and Technology Publications (2017). https://doi.org/10.5220/0006412700730082
56. Prabhu, S.M., Mosterman, P.J.: Model-based design of a power window system: modeling, simulation and validation. In: Proceedings of IMAC-XXII: a Conference on Structural Dynamics, Society for Experimental Mechanics Inc, Dearborn, MI (2004)
57. Rowson, J.A.: Hardware/Software co-simulation. In: 31st Conference on Design Automation, pp. 439–440 (1994)
58. Schweiger, G., Engel, G., Schoeggl, J., Hafner, I., Gomes, C., Nouidui, T.: Co-simulation – an empirical survey: applications, recent developments and future challenges. In: MATHMOD 2018 Extended Abstract Volume, pp. 125–126. ARGESIM Publisher Vienna, Vienna, Austria (2018). https://www.argesim.org/publications/a55286
59. Schweiger, G., Gomes, C., Hafner, I., Engel, G., Nouidui, T.S., Popper, N., Schoggl, J.P.: Co-simulation: leveraging the potential of urban energy system simulation. EuroHeat Power **15**(I–II), 13–16 (2018)
60. Spiegel, M., Reynolds, P., Brogan, D.: A case study of model context for simulation composability and reusability. In: Proceedings of the Winter Simulation Conference, vol. 2005, pp. 437–444. IEEE (2005). http://ieeexplore.ieee.org/document/1574279/
61. Thule, C., Gomes, C., Deantoni, J., Larsen, P.G., Brauer, J., Vangheluwe, H.: Towards the Verification of Hybrid Co-simulation Algorithms. Submitted to CoSim-CPS (2018)
62. Tomiyama, T., D'Amelio, V., Urbanic, J., ElMaraghy, W.: Complexity of multidisciplinary design. CIRP Ann. Manufact. Technol. **56**(1), 185–188 (2007)
63. Uchitel, S., Yankelevich, D.: Enhancing architectural mismatch detection with assumptions. In: 2000 Seventh IEEE International Conference and Workshop on the Engineering of Computer Based Systems, (ECBS 2000) Proceedings, pp. 138–146 (2000)
64. Van Acker, B., Denil, J., Meulenaere, P.D., Vangheluwe, H.: Generation of an optimised master algorithm for FMI co-simulation. In: Barros, F., Wang, M.H., Prähofer, H., Hu, X. (eds.) Symposium on Theory of Modeling and Simulation-DEVS Integrative, pp. 946–953. Society for Computer Simulation International San Diego, CA, USA, Alexandria, Virginia, USA, April 2015
65. Van der Auweraer, H., Anthonis, J., De Bruyne, S., Leuridan, J.: Virtual engineering at work: the challenges for designing mechatronic products. Eng. Comput. **29**(3), 389–408 (2013)
66. Vangheluwe, H., De Lara, J., Mosterman, P.J.: An introduction to multi-paradigm modelling and simulation. In: AI, Simulation and Planning in High Autonomy Systems, pp. 9–20. SCS (2002)
67. Vangheluwe, H.L., Vansteenkiste, G.C., Kerckhoffs, E.J.: Simulation for the future: progress of the esprit basic research Working Group 8467. In: Proceedings of the 1996 European Simulation Symposium, pp. XXIX–XXXIV. Society for Computer Simulation International, Genoa (1996)

68. Wanner, G., Hairer, E.: Solving Ordinary Differential Equations I: Nonstiff Problems, vol. 1. Springer, Heidelberg (1991). https://doi.org/10.1007/978-3-540-78862-1. Springer s edn.
69. Wu, M.C., Shih, M.C.: Simulated and experimental study of hydraulic anti-lock braking system using sliding-mode PWM control. Mechatronics **13**(4), 331–351 (2003)
70. Zeigler, B.P.: Theory of Modelling and Simulation. Wiley, New York (1976)