# Foundations for Continuous Time Hierarchical Co-simulation

Cláudio Gomes
University of Antwerp
claudio.gomes@uantwerpen.be

## Abstract

Complex systems have to decomposed into sub-systems which are developed by specialized teams. Modeling and simulation techniques help each team achieve a locally optimal solution but they fail to help all teams achieve a globally optimal solution. This is due to each team having its own models made in its own tools, and external suppliers having intellectual property. The result is that it is difficult to simulate the coupled system.

Co-simulation is proposed as a way to enable such simulation. Simulators communicate and collaborate as black boxes. The technique has been used in a number of domains improving the overall cost of engineered systems. In these use cases, a minimum common denominator is assumed for the capabilities of simulators. Leveraging the optional capabilities of simulators improves the performance/accuracy of a co-simulation, but the number of decisions the orchestration mechanism has to make grows exponentially. Different, conflicting concerns have to be addressed in an optimal way.

We propose a way to deal with each concern independently and possibly reuse existing orchestration algorithms that perform better with respect to the conflicting concerns. Our approach leverages Model Driven Development techniques to process a co-simulation scenario into a canonical/trivial version, where fewer decisions remain to be made. Along the process, conflicting concerns will be addressed as an optimization problem, and an appropriate cost function identified. If successful, the result of this research allows co-simulation scenarios that may offer real-time guarantees, bounds in the maximum error made, or packet size in the communication between simulators.

## 1. Introduction

Integration – the interconnection of the components that comprise a system – is identified as a major source of problems [28, 31] in the concurrent development of complex engineered systems. It is usually caused by assumptions about other components of the system having to be made early in the development of each component.

Modeling and simulation techniques are used to mitigate these issues: models of components are created and simulated before any physical prototype is built. Simulation can also be used to analyze the behavior of interacting models of components, created with different languages [10]. The simulation of interactions between models specified in different languages is an open challenge [16], mostly done with a small number of simulators with similar features, making it difficult to generalize to other scenarios. To aggravate, specialized suppliers of components are interested in protecting their Intellectual Property (IP) leading to the situation were the simulation needs to be made without access to the models [6].

Co-simulation is a technique to couple multiple simulators, each simulating a single component, often seen as a black box, in order to simulate the whole system. An orchestration algorithm is responsible for ensuring the communication between the simulators. The lack of information about the simulators makes co-simulation a hard challenge which must be addressed in a systematic way. Despite this, co-simulation shortens development time and improves quality, as reported by the industrial partners of the DESTECS project [7, 11, 26]. For other application domains where co-simulation has been applied, see [19, 23, 25, 31], for example.

Recognizing the potential of co-simulation, the Functional Mockup Interface (FMI) standard [6] was created to provide a common interface for different simulation tools to communicate. Obviously, these tools have different capabilities and the FMI standard acts as a common denominator, requiring the basic features that enable co-simulation. Many other features are available in the state of the art. See Fig. 1 (a) for some of these features.

The optional capabilities of simulators make for a combinatorial explosion when these are coupled in a co-simulation, making the development of an orchestration mechanism a hard challenge, with many different, often conflicting, concerns. In this project, we address this

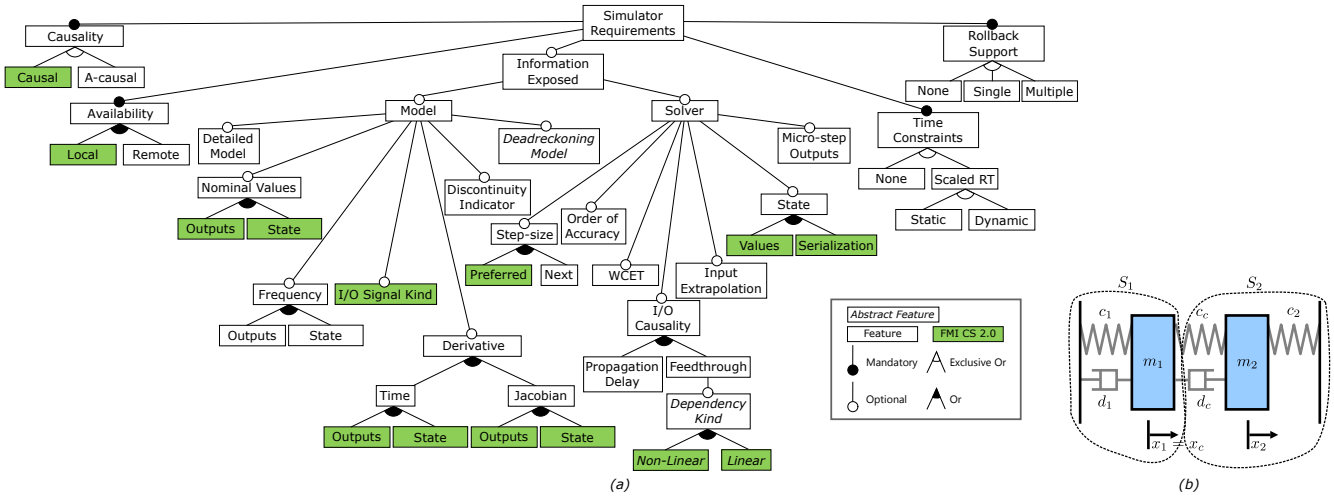


**Figure 1.** (a) Feature model mapping the capabilities of the simulators encountered in the state of the art. (b) A 2-DOF Oscillator as a coupling of two sub-systems.

challenge and we propose a way to take advantage of the optional features provided by simulators, instead of just using the common denominator, without resorting to a complex orchestration algorithm. This possibly allows for the reuse of existing orchestration algorithms while at the same time providing a better control over the accuracy/performance tradeoff.

## 2. Background

In order to properly define co-simulation scenarios, some background terms have to be introduced.

***Dynamical System*** A dynamical system is a model characterized by a state and a notion of evolution rules. An example is a mass-spring-damper system, depicted in Fig. 1 (b):

$$\begin{aligned} m_1 \cdot \ddot{x_1} &= -c_1 \cdot x_1 - d_1 \cdot \dot{x_1} + F_e \\ x_1(0) &= p_1; \quad \dot{x_1}(0) = s_1 \end{aligned} \tag{1}$$

where $c_1$ is the spring stiffness constant and $d_1$ the damping coefficient; $m_1$ is the mass; $p_1$ and $s_1$ the initial position and velocity; and $F_e$ denotes the external force over time acting on the mass. We consider dynamical systems that can be written in the state space form:

$$\dot{x} = f(x, u) \;\; ; \;\; x(0) = x_0 \tag{2}$$

where $x(t)$ is the state vector, $u(t)$ the input vector, and $x_0$ is the initial state.

***Behavior Trace*** The trajectory followed by the state over time is called the behavior trace of the dynamical system. Behavior traces can be exact (also called analytical) or approximations. In the above example, the function $x_1(t)$ that satisfies Eq. (1) is the behavior trace.

***Experimental Frame*** The experimental frame describes a set of assumptions in which the behavior trace of the dynamical system can be compared with the one of the original system [3, 29, 32, 33, 35].

***Validity*** In order to be used successfully as models of the systems, dynamical systems have to be valid within the experimental frame in which the they are defined. The validity of the dynamical system is then the difference between the behavior trace of the dynamical system and the behavior trace of the original system, measured under the assumptions of the experimental frame. For example, the Hooke's law in the mass-spring-damper system can only be used to predict the reaction force of the spring for small deformations.

***Solver*** A solver is an algorithm capable of obtaining the approximate behavior trace of a dynamical system. It is typically an iterative procedure that advances the simulated time and approximates the values of the variables at that point in time. For a dynamical system in the form of Eq. (2), the Forward Euler solver is given as:

$$\begin{aligned} \tilde{x}(t+h) &:= \tilde{x}(t) + f(\tilde{x}(t), u(t)) \cdot h \\ \tilde{x}(0) &:= x(0) \end{aligned} \tag{3}$$

where $\tilde{x}$ is the approximated state vector, $u(t)$ the input, and $h > 0$ is the micro-step size.

***Accuracy*** Accuracy is the difference between an approximate behavior trace and an exact one. In most practical cases, the correct behavior trace is difficult to obtain. However, it is possible to get a worst case estimate in the order of accuracy of a solver, provided that the system obeys certain, physically meaningful, assumptions (e.g., state continuity, Lipschitz conditions [2], and conservation laws [24]).

***Simulator*** The composition of a solver with a specific model is called a simulator. For example, to get a simulator of the mass-spring-damper system, write Eq. (1) in state space form, combine with Eq. (3) to get:

$$\begin{aligned}\tilde{x}_1(t+h_1) &:= \tilde{x}_1(t) + v_1(t)\cdot h_1 \\ \tilde{v}_1(t+h_1) &:= \tilde{v}_1(t) + \frac{(-c_1\tilde{x}_1(t) - d_1\tilde{v}_1(t) + F_e(t))}{m_1}\cdot h_1\end{aligned} \tag{4}$$

where $h_1$ is the micro-step size, $\tilde{x}_1(0) := p_1$, and $\tilde{v}_1(0) := s_1$

In general a simulator can be represented as:

$$\begin{aligned}&S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i}\rangle \\ &\delta_i : \mathbb{R}\times X_i \times U_i \to X_i \\ &\lambda_i : \mathbb{R}\times X_i \times U_i \to Y_i \text{ or } \mathbb{R}\times X_i \to Y_i \\ &x_i(0) \in X_i \\ &\phi_{U_i} : \mathbb{R}\times U_i \times \ldots \times U_i \to U_i\end{aligned} \tag{5}$$

where:
$X_i$ is the state set, typically $\mathbb{R}^n$; $U_i$ is the input set, typically $\mathbb{R}^m$; $Y_i$ is the output set, typically $\mathbb{R}^p$; $x_i(0)$ is the initial state; $\delta_i(t, x_i(t), u_i(t)) = x_i(t+H)$ is the function that instructs the simulator to compute a behavior trace from $t$ to $t+H$, making use of the input extrapolation function $\phi_{U_i}$; $H \in \mathbb{R}$ is the communication step size; and $\lambda_i(t, x_i(t), u_i(t)) = y_i(t)$ is the output function. The input extrapolation function $\phi_{U_i}$ plays an important role in guaranteeing that the simulator does not read values from the environment while computing the behavior trace in the interval $t \to t+H$. Oftentimes, constant extrapolation from the last known input is used, that is, $\phi_{U_i}(\tau, u_i(t)) = u_i(t)$, for $\tau \in [t, t+H]$.

***Co-simulation Scenario*** Simulators can have inputs and outputs, which capture the environment in which the original system operates. They can be combined by specifying how their inputs/outputs are connected. A co-simulation scenario is a specific arrangement of simulators and their I/O coupling conditions. An autonomous scenario requires at least the following information:

$$\begin{aligned}&CS = \langle S, L\rangle \\ &S = \{S_1, \ldots S_n\} \\ &L : Y_1 \times \ldots \times Y_n \times U_1 \times \ldots \times U_n \to \mathbb{R}^m\end{aligned} \tag{6}$$

$S$ is the set of causal simulators, each defined as in Eq. (5); and $L$ induces the following coupling constraint:

$$L(y_1, \ldots, y_n, u_1, \ldots, u_n) = \bar{0}$$

As an example, for the co-simulation scenario corresponding to the multi-body system of Fig. 1 (b), we have:

$$\begin{aligned}&CS = \langle \mathbb{R}, \{S_1, S_2\}, L\rangle \\ &L = \begin{bmatrix} x_c - v_1 \\ \dot{x}_c - x_1 \\ F_e - F_c \end{bmatrix}\end{aligned} \tag{7}$$

where:
$S_1$ is the simulator defined in Eq. (4) and the definition of $S_2$ is omitted; $x_c$, $\dot{x}_c$ are the inputs of $S_2$, and $F_e$ is the input of $S_1$; $x_1$, $v_1$ are outputs of $S_1$ and $F_c$ is the output of $S_2$;

***Trivial Co-simulation Scenario*** A co-simulation scenario is trivial when the coupling conditions can be transformed into a set of assignments from outputs to inputs. To achieve this: (1) no input/output is a function of itself; (2) for each input, there is an output that provides its value. The co-simulation scenario described by Eq. (7) is trivial.

***Orchestration Algorithm*** Given a co-simulation scenario, an orchestration algorithm coordinates the simulators ensuring that each progresses in time and receives inputs. A trivial scenario can be simulated with Algorithm 1.

**ALGORITHM 1:** Generic orchestration mechanism for trivial co-simulation scenarios.

**Data:** Stop time $T_f$, a co-simulation scenario $\langle S, L\rangle$, and a communication step size $H$.
**Result:** A co-simulation trace.
$t := 0$;
**while** $t < T_f$ **do**
  Solve the following system for the unknowns $y_1(t), \ldots, y_n(t), u_1(t), \ldots, u_n(t)$:
  $\begin{cases} y_i(t) = \lambda_i(t, x_i(t), u_i(t)), \text{ for } i = 1, \ldots, n \\ L(y_1(t), \ldots, y_n, u_1(t), \ldots, u_n(t)) = \bar{0} \end{cases}$;
  The values $[y_1(t), \ldots, y_n(t), u_1(t), \ldots, u_n(t)]^T$ denote a point at time $t$ of the co-simulation trace;
  Instruct each simulator to advance to the next communication step:
  $x_i(t+H) := \delta_i(t, H, x_i(t), u_i(t)), \text{ for } i = 1, \ldots, n$;
  Advance time:
  $t := t + H$;
**end**

***Hierarchical Co-simulation*** A co-simulator is obtained when an orchestration mechanism is coupled with a co-simulation scenario. According to our nomenclature, a co-simulator is a simulator and can be specified as in Eq. (5). This means that a co-simulation scenario can be comprised of simulators, which can themselves be co-simulation scenarios with suitable orchestrators. This is important because hierarchical systems are best described by hierarchical co-simulation scenarios.

In the following sections, we describe non-trivial co-simulation scenarios and, instead of describing how these can be solved with more complex orchestration

algorithms, we describe how they can be translated into trivial co-simulation scenarios. This approach, supported by modeling the co-simulation scenarios, allows for clear separation of concerns and provides flexibility in choosing how to deal with each of them.

## 3. Concerns in Co-simulation

### 3.1 Accuracy Concern

The accuracy of a co-simulation trace is the degree to which it conforms to the real trace. Error – the difference between the co-simulation trace and the real trace – is then a measure of accuracy. Obtaining the real trace, for most dynamical systems, is currently impossible. However, there is an important result in simulation – convergence – which allows the order of the worst case deviation from the real trace made by a numerical method to be controlled by adjusting the micro-step size $h_i$ of the solver. The same result can be applied to certain co-simulation scenarios [2], allowing the communication step size $H$ to control the global error.

Adjusting the communication step size $H$ is then an accuracy concern. Given a co-simulation scenario, $H$ can be controlled by an extra simulator $S_H$, introduced artificially, whose outputs are the time variable $t$ and $H$, and inputs are relevant outputs of other simulators. A new independent variable $s$ with a communication step size of 1 is introduced. Variables $t$ and $H$ become functions of $s$. A similar translation has been proposed in [22] and the simulator $S_H$ can implement a well known PI-Controller. See [2, 9, 13, 24, 34] for error control alternatives in co-simulation. Note that these can also be applied in our approach because $S_H$ can be a co-simulation scenario (e.g., a copy of the original scenario running at a communication step size of $\frac{H}{2}$, for Richardson extrapolation). Fig. 2 (a) summarizes this approach.

### 3.2 Algebraic Loops Concern

Algebraic loops occur whenever there is a variable that is a function of itself. The state and output of each simulator $S_i$ in a co-simulation can be written as:

$$\begin{aligned} x_i(t+H) &= \delta_i(t, H, x_i(t), u_i(t)) \\ y_i(t+H) &= \lambda_i(t, x_i(t+H), u_i(t+H)) \end{aligned} \tag{8}$$

Taking into account the coupling conditions, it is easy to see that an output of a simulator may depend on itself. These kinds of algebraic loops in the output equations can be avoided by replacing $u_i(t+H)$ in Eq. (8) by the corresponding extrapolation $\phi_{u_i}(H, u_i(n \cdot H), u_i((n-1) \cdot H), \ldots)$ which does not depend on $u_i((n+1) \cdot H)$, thus breaking the algebraic loop [1]. However, as is shown in [2, 18], and empirically in [4], breaking an algebraic loop instead of solving it can lead to a high error in the co-simulation. A better way is to use a fixed point iteration technique, where in the general case, simulators are asked to compute the interval $t \to t+H$ many times, with improved inputs, until some convergence criteria is met.

Given a co-simulation scenario with algebraic loops, extra information is necessary to be able to identify the loops, as pointed out in [2, 8, 30]. Assuming that this information exists, the simulators that are involved in an algebraic loop can be "lifted out" and replaced by a single simulator $S_{SC}$ whose $\delta_{SC}$ implements the iteration techniques that solves the loop. The result is a trivial scenario that can be simulated by the orchestration mechanism of Algorithm 1. This adaptation is summarized in Fig. 2 (b)

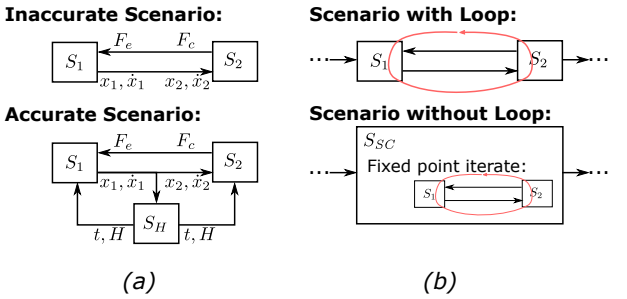


**Figure 2.** (a) Transformation that deals with accuracy concern. (b) Dealing with algebraic loop concern.

### 3.3 Communication Concern

If simulators in a co-simulation scenario execute in different computers, a small $H$ incurs a too high communication cost. On the other hand, using a large $H$ places the burden in the functions $\phi$ to accurately extrapolate the inputs of each simulator across a large interval. In many cases – in particular, for the FMI Standard –, each simulator $S_i$ is the one responsible for implementing $\phi_{U_i}$. Therefore a problem exists where $H$ should be high to reduce the communication cost, but the accuracy of functions $\phi$ cannot be improved to compensate.

To show how this problem can be addressed, consider the scenario shown in Fig. 3 (a), where the simulators communicate every $H$ units of time. The purpose is to increase $H$ and mitigate the accuracy loss. For that, replace each group of simulators in the same computer by a single simulator. Then, the new simulator encapsulates a co-simulation scenario where the internal communication step size is $H_{\text{small}} < H$. An artificial simulator is introduced to provide approximated values of the outputs of the simulators in the other computers. These can be extrapolations from the values collected at the other computers. In the example, $S_2'$ collects the

[1] See [18] for the other kind of algebraic loops.

outputs of $S_1$ and sends them over the wire to $S_1'$ at every $H$ time units. The smaller $H_{\text{small}}$, the finer grained the extrapolation of $S_1$ will be.

This approach can be applied whenever an input extrapolation function needs to be provided, regardless of the computer in which the simulators execute. For instance, when the scenario is comprised of simulators whose outputs evolve at very different rates, as happens in circuit simulation [21], better extrapolation functions can be provided to save computation on the "slow" components. Furthermore, if simulators provide rollback capabilities, an iterative predictor correct method can be made, yielding a generalized waveform relaxation iteration [20].

### 3.4 Modularity Concern

It is possible that, even without algebraic loops, the coupling conditions do not yield a set of assignments. To show how this can happen, consider the co-simulation scenario that represents the coupled system on top of Fig. 3 (b). The input to the first simulator is the external force $F_e$ and the outputs are $[\tilde{x}_1, \tilde{v}_1]^T$ (see Eq. (4)). The input to the second simulator is the external force $F_c$ and the outputs are $[\tilde{x}_3, \tilde{v}_3]^T$. Clearly, there is a mismatch: the outputs $[\tilde{x}_1, \tilde{v}_1]^T$ of the first simulator cannot be coupled directly to the input $F_c$ of the second simulator, and vice versa. However, the massless link restricts the outputs of the two sub-systems to be the same and $F_e = F_c$, whatever that force may be.

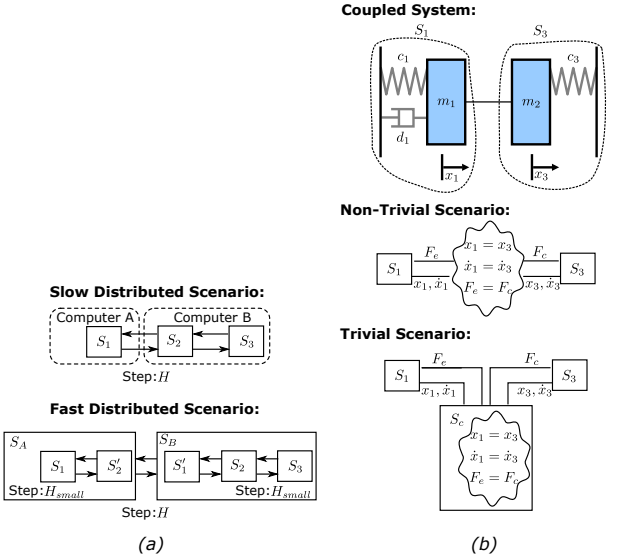


**Figure 3.** (a) Dealing with distribution concern. (b) Transformation that solves causality conflicts.

Our solution to this concern is similar to that of [14] and is summarized in Fig. 3 (b). The essence is to add an artificial simulator to the co-simulation, which calculates the appropriate inputs to the simulators, that ensure equal outputs. For the details of how those calculations can be done, see [1, 15, 27].

## 4. Related Work

The aim of this work is to generalize the work done by Van Acker et al. [30], where a language is proposed to configure the co-simulation scenario with extra information identifying the optimal rates for each simulator and algebraic loops. In our work, we recognize that, for each concern, there are multiple solutions, with differing orders of "cost", that depend on the sensitivities between simulators. The work in Kajtazovic et al. [17] is similar to ours in the sense that a generative approach is followed, but there is no focus into identifying and solving the multiple concerns involved in devising an orchestration algorithm. The work in Benedikt and Holzinger [5] presents some initial steps toward an orchestration mechanism that adapts at run-time. Similarly to our work, it recognizes that the orchestration mechanism is highly dependent on the co-simulation scenario and that it should be tuned automatically. However, we differ in the approach: we do it statically, as opposed to at run-time, like they do.

## 5. Conclusion

This project aims at dealing separately with the many concerns that originate in continuous time co-simulation. Our approach is to stick to a simple orchestration algorithm, and transform the scenarios, by introducing artificial simulators. Fig. 4 summarizes our overall approach.

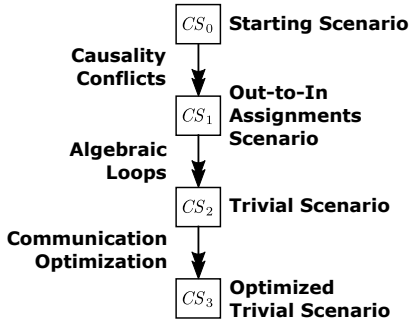


**Figure 4.** Overview of the main transformation stages.

The advantage is that there is a clear set of pre-conditions and post-conditions for each transformation, showing the separation of concerns. Based on anecdotal evidence, we propose the order of the stages to be the one in the figure. This order ensures that no concern resurfaces in later stages of the transformation. Because of the performance/accuracy tradeoff, the communication concern can only be addressed as an optimization problem and we allow for the application of optimization techniques to find an optimal co-simulation sce-

nario. One disadvantage of our approach is that co-simulation scenario can quickly become unreadable, due to the injected artificial simulators. Further evaluation is necessary to measure the how complex non-trivial co-simulation scenarios can become after being transformed.

The co-simulation scenarios used in the current work were created artificially. In the future, we aim at testing these approaches with real co-simulation scenarios, such as the ones developed in [12] and in the INTO-CPS project [2]. Furthermore, the order of convergence has to be studied for the approach described in Section 3.3.

## Acknowledgment

This work was partially funded with PhD fellowship from the Agency for Innovation by Science and Technology in Flanders (IWT). The author is grateful for all the pertinent comments provided by the reviewers and other contestants of the Student Research Competition.

## References

[1] M. Arnold and M. Günther. Preconditioned Dynamic Iteration for Coupled Differential-Algebraic Systems. *BIT Numerical Mathematics*, 41(1):1–25, 2001. doi: 10.1023/A:1021909032551.

[2] M. Arnold, C. Clauß, and T. Schierz. Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0. In S. Schöps, A. Bartel, M. Günther, W. E. J. ter Maten, and C. P. Müller, editors, *Progress in Differential-Algebraic Equations*, pages 107–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44926-4. doi: 10.1007/978-3-662-44926-4_6.

[3] P. I. Barton and C. C. Pantelides. Modeling of combined discrete/continuous processes. *AIChE Journal*, 40(6): 966–979, jun 1994. ISSN 0001-1541. doi: 10.1002/aic.690400608.

[4] J. Bastian, C. Clauß, S. Wolf, and P. Schneider. Master for Co-Simulation Using FMI. In *8th International Modelica Conference*, pages 115–120, Dresden, Germany, jun 2011. Fraunhofer Institute for Integrated Circuits IIS. doi: 10.3384/ecp11063115.

[5] M. Benedikt and F. R. Holzinger. Automated configuration for non-iterative co-simulation. In *17th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, pages 1–7, Montpellier, apr 2016. IEEE. ISBN 978-1-5090-2106-2. doi: 10.1109/EuroSimE.2016.7463355.

[6] T. Blockwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International MODELICA Conference*, pages 173–184, Munich, Germany, nov 2012. Linköping University Electronic Press; Linköpings universitet. doi: 10.3384/ecp12076173.

[7] J. F. Broenink and Y. Ni. Model-driven robot-software design using integrated models and co-simulation. In *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pages 339–344, 2012. ISBN VO -. doi: 10.1109/SAMOS.2012.6404197.

[8] D. Broman, C. Brooks, L. Greenberg, E. A. Lee, M. Masin, S. Tripakis, and M. Wetter. Determinate composition of FMUs for co-simulation. In *Eleventh ACM International Conference on Embedded Software*, Montreal, Quebec, Canada, 2013. IEEE Press Piscataway, NJ, USA. ISBN 978-1-4799-1443-2.

[9] M. Busch. Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 96(9):1061–1081, sep 2016. ISSN 00442267. doi: 10.1002/zamm.201500196.

[10] C. W. De Silva. *Mechatronics: an integrated approach*. CRC press, 2004.

[11] U. Eliasson, R. Heldal, J. Lantz, and C. Berger. Agile Model-Driven Engineering in Mechatronic Systems - An Industrial Case Study. In J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, editors, *Model-Driven Engineering Languages and Systems SE - 27*, volume 8767 of *Lecture Notes in Computer Science*, pages 433–449. Springer International Publishing, 2014. ISBN 978-3-319-11652-5. doi: 10.1007/978-3-319-11653-2_27.

[12] J. Fitzgerald and K. Pierce. *Collaborative Design for Embedded Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-54117-9. doi: 10.1007/978-3-642-54118-6.

[13] V. Galtier, G. Plessis, and L. Renardi. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Spring Simulation Multi-Conference*, pages 804–811. Society for Computer Simulation International, 2015.

[14] B. Gu and H. H. Asada. Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference, 2001. Proceedings of the 2001*, volume 3, pages 2273–2278 vol.3, 2001. ISBN 0743-1619 VO - 3. doi: 10.1109/ACC.2001.946089.

[15] B. Gu and H. H. Asada. Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models. *Journal of Dynamic Systems, Measurement, and Control*, 126(1):1, apr 2004. ISSN 00220434. doi: 10.1115/1.1648307.

[16] C. Hardebolle and F. Boulanger. Exploring Multi-Paradigm Modeling Techniques. *SIMULATION*, 85(11-12):688–708, nov 2009. doi: 10.1177/0037549709105240.

[17] S. Kajtazovic, C. Steger, A. Schuhai, and M. Pistauer. Automatic Generation of a Coverification Platform. In A. Vachoux, editor, *Applications of Specification and*

[2] `http://into-cps.au.dk/`

*Design Languages for SoCs*, pages 187–203. Springer Netherlands, Dordrecht, 2006. ISBN 978-1-4020-4997-2. doi: 10.1007/978-1-4020-4998-9_11.

[18] R. Kübler and W. Schiehlen. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems*, 6(2):93–113, jun 2000. ISSN 1387-3954. doi: 10.1076/1387-3954(200006)6:2;1-M;FT093.

[19] M. Kudelski, L. M. Gambardella, and G. A. Di Caro. RoboNetSim: An integrated framework for multi-robot and network simulation. *Robotics and Autonomous Systems*, 61(5):483–496, may 2013. ISSN 09218890. doi: 10.1016/j.robot.2013.01.003.

[20] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits. In *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, volume 1, pages 131–145, 1982. ISBN 0278-0070 VO - 1. doi: 10.1109/TCAD.1982.1270004.

[21] W. J. McCalla. *Fundamentals of computer-aided circuit simulation*, volume 37. Springer Science & Business Media, 1987. ISBN 1461320119.

[22] P. J. Mosterman, J. Zander, G. Hamon, and B. Denckla. A computational model of time for stiff hybrid systems applied to control synthesis. *Control Engineering Practice*, 20(1):2–13, 2012. ISSN 09670661. doi: 10.1016/j.conengprac.2011.04.013.

[23] N. Pedersen, T. Bojsen, J. Madsen, and M. Vejlgaard-Laursen. FMI for Co-Simulation of Embedded Control Software. In *The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan*, number 124, pages 70–77. Linköping University Electronic Press, may 2016. ISBN 1650-3740. doi: 10.3384/ecp1612470.

[24] S. Sadjina, L. T. Kyllingstad, E. Pedersen, and S. Skjong. Energy Conservation and Power Bonds in Co-Simulations: Non-Iterative Adaptive Step Size Control and Error Estimation. *arXiv preprint arXiv:1602.06434*, 2016.

[25] R. A. Saleh, S.-J. Jou, and A. R. Newton. *Mixed-mode simulation and analog multilevel simulation*, volume 279. Springer Science & Business Media, 2013. ISBN 1475758545.

[26] S.-A. Schneider, J. Frimberger, and M. Folie. Significant Reduction of Validation Efforts for Dynamic Light Functions with FMI for Multi-Domain Integration and Test Platforms. In *10th International Modelica Conference*, 2014.

[27] B. Schweizer, D. Lu, and P. Li. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. *Multibody System Dynamics*, 36(1):1–36, jan 2016. ISSN 1384-5640. doi: 10.1007/s11044-015-9464-9.

[28] T. Tomiyama, V. D'Amelio, J. Urbanic, and W. ElMaraghy. Complexity of Multi-Disciplinary Design. *CIRP Annals - Manufacturing Technology*, 56(1):185–188, 2007. ISSN 00078506. doi: 10.1016/j.cirp.2007.05.044.

[29] M. K. Traoré and A. Muzy. Capturing the dual relationship between simulation models and their context. *Simulation Modelling Practice and Theory*, 14(2):126–142, feb 2006. ISSN 1569190X. doi: 10.1016/j.simpat.2005.03.002.

[30] B. Van Acker, J. Denil, P. D. Meulenaere, H. Vangheluwe, B. Vanacker, and P. Demeulenaere. Generation of an Optimised Master Algorithm for FMI Co-simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative*, pages 946–953. Society for Computer Simulation International, 2015.

[31] H. Van der Auweraer, J. Anthonis, S. De Bruyne, and J. Leuridan. Virtual engineering at work: the challenges for designing mechatronic products. *Engineering with Computers*, 29(3):389–408, 2013. ISSN 0177-0667. doi: 10.1007/s00366-012-0286-6.

[32] H. Vangheluwe. Foundations of Modelling and Simulation of Complex Systems. *EASST*, 10, 2008. doi: 10.14279/tuj.eceasst.10.162.148.

[33] H. Vangheluwe, J. De Lara, and P. J. Mosterman. An introduction to multi-paradigm modelling and simulation. In *Proceedings of AIS2002 (AI, Simulation & Planning)*, pages 9–20. SCS, 2002.

[34] A. Verhoeven, B. Tasic, T. G. J. Beelen, E. J. W. ter Maten, and R. M. M. Mattheij. BDF compound-fast multirate transient analysis with adaptive stepsize control. *J. Numer. Anal. Ind. Appl. Math*, 3(3-4):275–297, 2008.

[35] B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press, 2 edition, 2000.