Claudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-Simulation: A Survey. In ACM Computing Surveys (CSUR), Volume 51, Issue 3, Article No. 49, ACM, 2018.
https://doi.org/10.1145/3179993

Co-simulation: a Survey

CLÁUDIO GOMES, University of Antwerp, Belgium
CASPER THULE, Aarhus University, Denmark
DAVID BROMAN, KTH Royal Institute of Technology, Sweden
PETER GORM LARSEN, Aarhus University, Denmark
HANS VANGHELUWE, University of Antwerp, Flanders Make, Belgium and McGill University, Canada

Modeling and simulation techniques are today extensively used both in industry and science. Parts of larger systems are, however, typically modeled and simulated by different techniques, tools, and algorithms. In addition, experts from different disciplines use various modeling and simulation techniques. Both these facts make it difficult to study coupled heterogeneous systems.

Co-simulation is an emerging enabling technique, where global simulation of a coupled system can be achieved by composing the simulations of its parts. Due to its potential and interdisciplinary nature, co-simulation is being studied in different disciplines but with limited sharing of findings.

In this survey, we study and survey the state-of-the-art techniques for co-simulation, with the goal of enhancing future research and highlighting the main challenges.

To study this broad topic, we start by focusing on discrete-event-based co-simulation, followed by continuous-time-based co-simulation. Finally, we explore the interactions be these two paradigms, in hybrid co-simulation.

This research was partially supported by Flanders Make vzw, the strategic research centre for the manufacturing industry, and a PhD fellowship grant from the Agency for Innovation by Science and Technology in Flanders (IWT). In addition, the work presented here is partially supported by the INTO-CPS project funded by the European Commission's Horizon 2020 programme under grant agreement number 664047. This work is also financially supported by the Swedish Foundation for Strategic Research (project FFL15-0032). We warmly thank the reviewers for their in-depth suggestions on how to improve this work.
Authors' addresses: Cláudio Gomes, Hans Vangheluwe, Department of Mathematics and Computer Science, University of Antwerp, Campus Middelheim, {M.G.028, M.G.116}, Middelheimlaan 1, 2020 Antwerpen, Belgium; email: {claudio.gomes,hans.vangheluwe}@uantwerp.be; Casper Thule, Peter Gorm Larsen, Department of Engineering, Aarhus University, Finlandsgade 22, DK-8200 Aarhus N, Denmark; email: {casper.thule,pgl}@eng.au.dk; David Broman, ICT/SCS, KTH Royal Institute of Technology, Kistagången 16, 164 40 Kista, Sweden; email: dbro@kth.se.
Authors' addresses: Cláudio Gomes, claudio.gomes@uantwerp.be, University of Antwerp, Department of Mathematics and Computer Science, Antwerpen, Middelheimlaan 1, 2000, Belgium; Casper Thule, casper.thule@eng.au.dk, Aarhus University, Department of Engineering, Aarhus, Finlandsgade 22, DK-8200, Denmark; David Broman, dbro@kth.se, KTH Royal Institute of Technology, ICT/SCS, Kista, Kistagången 16, 164 40, Sweden; Peter Gorm Larsen, pgl@eng.au.dk, Aarhus University, Department of Engineering, Aarhus, Finlandsgade 22, DK-8200, Denmark; Hans Vangheluwe, hans.vangheluwe@uantwerp.be, University of Antwerp, Flanders Make, Department of Mathematics and Computer Science, Antwerpen, Middelheimlaan 1, 2000, Belgium, McGill University, School of Computer Science, Montréal, Québec, 3480 University Street, H3A 0E9, Canada.

0360-0300/2018/1-ART1 $15.00
https://doi.org/10.1145/3179993

To survey the current techniques, tools, and research challenges, we systematically classify recently published research literature on co-simulation, and summarize it into a taxonomy. As a result, we identify the need for finding generic approaches for modular, stable, and accurate, coupling of simulation units, as well as expressing the adaptations required to ensure that the coupling is correct.

CCS Concepts: • **Computing methodologies** → **Discrete-event simulation**; **Continuous simulation**; *Systems theory*; *Agent / discrete models*; *Continuous models*; *Simulation support systems*;

Additional Key Words and Phrases: Co-simulation, Simulation, Compositionality

ACM Reference Format:
Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. 2018. Co-simulation: a Survey. *ACM Comput. Surv.* 1, 1, Article 1 (January 2018), 35 pages. https://doi.org/10.1145/3179993

1 INTRODUCTION

Truly complex engineered systems that integrate physical, software, and network aspects are emerging [110][46], posing challenges in their design, operation, and maintenance.

The design of such systems, due to market pressure, has to be concurrent and distributed, that is, divided between different teams and/or external suppliers, each in its own domain and each with its own tools. Each participant develops a partial solution to a constituent system that needs to be integrated with all the other partial solutions. The later in the process the integration is done, the less optimal it is [59][127].

Innovative and optimal multi-disciplinary solutions can only be achieved through an holistic development process [61] where the partial solutions developed independently are integrated sooner and more frequently, as each solution is refined. Furthermore, the traditional activities carried out at the partial solution level—such as requirements compliance check, or design space exploration—can be repeated at the global level, and salient properties spanning multiple constituent systems can be studied.

Modeling and simulation can improve the development of the partial solutions (e.g., see [94][52]), but falls short in fostering this holistic development process [71]. To understand why, one has to observe that: (i) models of each partial solution cannot be exchanged or integrated easily, because these are likely developed by one of the many specialized tools deployed over the past 20 years; (ii) externally supplied models may have Intellectual Property (IP) that cannot be cheaply disclosed to system integrators; and (iii) as solutions are refined, the system should be evaluated by integrating physical prototypes, software components, and even human operators, in what are denoted as Model/Software/Hardware/Human-in-the-loop simulations [13][45].

Consider now the interaction with, or operation of, a complex system. Such operation requires training, which, for safety or costs, may have to be conducted in a virtual environment. Developing a virtual environment is a difficult task [3] and reusing the models used in the development of the system allows the bulk of the effort to be redirected to where it is essential. Again, due to the aforementioned reasons, it may be difficult to obtain a single model of the whole system.

A high fidelity model of a system can also be used for maintenance of the system. Advanced sensory information, collected during the normal operation of the system, can be fed into a simulator to predict and prevent faults [99].

These are but a small sample of reasons for (and advantages of) being able to accurately compute the behavior of a coupled system. The fact that it should be carried out from

a collection of interacting behaviors of the individual parts is what makes it a difficult challenge.

Co-simulation consists of the theory and techniques to enable global simulation of a coupled system via the composition of simulators. 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 [1], software and its execution platform [30], dedicated real-time hardware simulators (e.g., [105]), physical test stands (e.g., [65, Fig. 3]), or human operators (e.g., [28, Fig. 24],[124, Fig. 6]).

From 2011 to 2016, there has been at least 48 reported industrial applications of co-simulation (see [152] and Section 6.3.1 for an example application). While many different engineering domains have benefited from the technique, most reports describe the coupling of two simulators, each a *mock-up* of a constituent system from a different domain. This unexplored potential is recognized in a number of completed and ongoing projects that address co-simulation (MODELISAR[1], DESTECS[2], INTO-CPS[3], ACOSAR[4], and ACoRTA[5]).

Contribution. We present a survey and a taxonomy, focused on the enabling techniques of co-simulation, as an attempt to bridge, relate, and classify the many approaches in the state of the art. Despite the growing interest in the benefits and scientific challenges of co-simulation, to the best of our knowledge, no existing survey attempts to cover the heterogeneous communities in which it is being studied. The lack of such a survey means that the same techniques are being proposed independently with limited sharing of findings. To give an example, the use of dead-reckoning models is a well known technique in discrete event co-simulation [42], but only very recently it was used in a continuous time co-simulation approach [129]. **Our objective** is to facilitate the exchange of solutions and techniques, highlight the essential challenges, and attain a deeper understanding of co-simulation.

To help structure the characteristics of the simulators and how they interact, we distinguish two main approaches for co-simulation: Discrete Event (DE), described in Section 3, and Continuous Time (CT), described in Section 4. Both of these can be, and are, used for the co-simulation of continuous, discrete, or hybrid coupled systems. We call Hybrid co-simulation, described in Section 5, a co-simulation approach that mixes the DE and CT approaches [6]. Section 6 summarizes the features provided by co-simulation frameworks, and classifies the state of the art with that taxonomy. Finally, Section 7 concludes this publication. The section below provides the terminology used in the rest of the survey.

2 MODELING, SIMULATION, AND CO-SIMULATION

A *dynamical system* is a model of a real system (for instance a physical system or a computer system) characterized by a state and a notion of evolution rules. For instance, a traffic light system can be modeled as a dynamical system that can be in one of four possible states (`red`, `yellow`, `green`, or `off`). The evolution rules may dictate that it changes from `red` to `green` after some time (e.g., 60 seconds). Another example is a mass-spring-damper, modeled by a set of first order Ordinary Differential Equations (ODEs).

[1] https://itea3.org/project/modelisar.html
[2] http://www.destecs.org/
[3] http://into-cps.au.dk/
[4] https://itea3.org/project/acosar.html
[5] http://www.v2c2.at/research/ee-software/projects/acorta/
[6] Note that in this survey, we are focusing on timed formalisms (also called models of computation). Other formalisms, with no or only logical notion of time, are not discussed in this survey. For an overview of formalisms and models of computation, see [7] and [81].

The *behavior trace* is the set of trajectories followed by the state (and outputs) of a dynamical system. For example, a state trajectory x can be defined as a mapping between a time base T and the set of reals $\mathbb{R}$, that is, $x : T \rightarrow \mathbb{R}$.

We refer to the time variable $t \in T$ as *simulated time*—or simply *time*, when no ambiguity exists—defined over a time base T (typical the real numbers $\mathbb{R}$), as opposed to the *wall-clock time* $\tau \in WcT$, which is the time that passes in the real world [95]. When computing the behavior trace of a dynamical system over an interval $[0, t]$ of simulated time, a computer takes τ units of wall-clock time that depend on t. τ can therefore be used to measure the run-time performance of simulators. Fig. 1a highlights different kinds of simulation, based on the relationship between τ and t. In *real-time simulation*, this relationship should be $t = \alpha\tau$, but enforcing $\alpha = 1$ is one of the main challenges in real-time simulation, and by extension, of co-simulation. Simulation tools that offer interactive visualization allow the user to pause the simulation and/or set a different value for α.

Knowing when a dynamical system can be used to predict the behavior of a real system is crucial. The *experimental frame* describes, in an abstract way, a set of assumptions in which the behavior trace of the dynamical system can be compared with the one of the real system [11][18][62][141][60]. By real system we mean either an existing physical system, or a system that does not yet exist. *Validity* is then the difference between the behavior trace of the dynamical system and the behavior trace of the real system, measured under the assumptions specified by the experimental frame. This is what conveys predictive power to dynamical systems. As examples, consider the small deformation assumption for Hooke's law, or the instantaneous transitions of state in the traffic light, both valid models (to some degree) of the corresponding real systems.

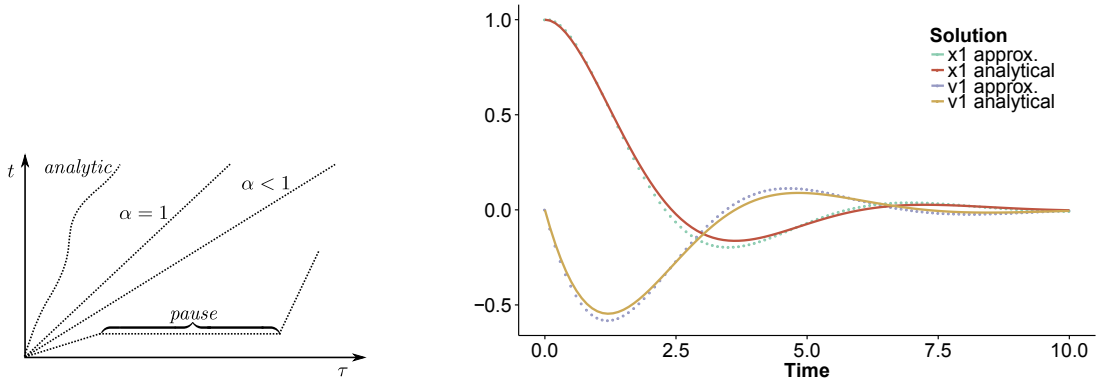
There are two generally accepted ways of obtaining the behavior trace of a dynamical system: translational (e.g., obtaining the analytical solution of an ODE) and operational (e.g., using a simulator to approximate the solution of an ODE). We focus on the latter.

A *simulator* (or solver) is an algorithm that computes the behavior trace of a dynamical system. If running in a digital computer, it is often the case that a simulator will only be able to approximate that trace. Two aspects contribute to the error in these approximations: inability to calculate a trajectory over a continuum, and the finite representation of infinitely small quantities. Fig. 1b shows an example approximation (dashed line) of the behavior trace (solid line) of the mass-spring-damper system, computed by the forward Euler method. Clearly, the trajectories differ.

In order to define what an *accurate simulator* is, or even be able to talk about error, we need to postulate that every dynamical system has an analytical behavior trace. The error can then be defined as the norm of the difference between the behavior trace produced by a simulator and the analytical trace. A simulator is accurate when the error is below a given threshold. Even if it is not possible to obtain the analytical behavior of every dynamical system, there are theoretical results that allow simulators to control the error they make. These techniques are applied to co-simulation in Section 4.3. In short, validity is a property of a dynamical system whereas accuracy is a property of a simulator [1].

In strict terms, a simulator is not readily executable: it needs a dynamical system and input trajectories, before being able to compute the behavior trace.

We use the term *simulation unit (SU)* to denote something that produces a behavior trace, when inputs are provided. A SU can be a composition of a simulator and a dynamical system, or it can be a real-world entity (with appropriate interface). Notice that, in contrast to a simulator, a SU only requires inputs to produce behavior.



(a) Classification of time constraints in simulation. Based on [118][137].

(b) Approximate behavior trace of the mass-spring-damper system. Parameters are: $m = 1 = p = c_1 = d_1 = 1$, and $s = F_e(t) = 0$. x is the displacement and v is the velocity.

Fig. 1

A *simulation* is the behavior trace obtained with a SU. The correctness of a SU is dictated by the correctness of the simulation, which depends on the accuracy of the simulator and the validity of the dynamical system.

As described in Section 1, it is useful to obtain correct simulations of complex, not yet existing, systems as a combination of the behaviors of its constituent parts. Suppose each part is represented by a SU. Then these can be coupled via their inputs/outputs to produce a behavior trace of the coupled system. A *co-simulation*, a special kind of simulation, is the collection of combined simulations produced by the coupled SUs.

The SUs are independent black boxes. Hence, an *orchestrator* is necessary to couple them. The orchestrator controls how the simulated time progresses in each SU and moves data from outputs to inputs according to a co-simulation scenario. A *co-simulation scenario* is the information necessary to ensure that a correct co-simulation can be obtained. It includes how the inputs of each SU are computed from outputs, their experimental frames, etc.

Analogously to the simulator and SU concepts, the composition of a specific orchestrator with a co-simulation scenario, yields a *co-SU*, which is a special kind of SU, and a substitute of the real coupled system. It follows that a co-simulation is the simulation trace computed by a co-SU. This characterization enables hierarchical co-simulation scenarios, where co-SUs are coupled.

In this survey, we focus on the coupling techniques of black box SUs, where limited knowledge of the models and simulators is available. However, as will become clear in the following sections, the black box restriction has to be relaxed so that certain properties related to correctness can be ensured. Understanding what kind of information should be revealed and how IP can still be protected is an active area of research in co-simulation.

Most challenges in co-simulation are related to compositionality: if every SU S_i in a co-simulation scenario satisfies property P, then the co-SU, with a suitable orchestrator, must also satisfy P. The correctness is a property that should be compositional in co-simulation. Other properties include validity, or accuracy. It is an open research question to ensure that a co-simulator is compositional for a given set of properties. The following three sections provide an overview of the information and techniques being used throughout the state of the art, divided into three main approaches: discrete event (Section 3), continuous time (Section 4), and hybrid (Section 5) co-simulation.

3 DISCRETE-EVENT-BASED CO-SIMULATION

The Discrete-Event-(DE)-based co-simulation approach describes a family of orchestrators and characteristics of simulation units (SUs) that are borrowed from the DE system simulation domain. We start with a description of DE systems, and then we extract the main concepts that characterize DE based co-simulation.

The traffic light is a good example of a DE system. It can be in one of the possible modes: `red`, `yellow`, `green`, or `off`. Initially, the traffic light can be red. Then, after 60 seconds, it changes to green. Alternatively, before those 60 seconds pass, some external entity (e.g., a police officer) may trigger a change from red to off. The output of this system can be an event signaling its change to a new color. This example captures some of the essential characteristics of a DE dynamical system: *reactivity* – instant reaction to external stimuli (turning off by an external entity); and *transiency* – a DE system can change its state multiple times in the same simulated time point, and receive simultaneous stimuli. In the traffic light, *transiency* would happen if the light changes always after 0s (instead of 60s), or if the police officer would turn off and on the traffic light in the same instant.

These characteristics are embraced in DE based co-simulation, where the orchestrator acknowledges that SUs can change their the internal state and exchange values despite the fact that the simulated time is stopped.

3.1 DE Simulation Units

A DE SU is a black box that exhibits the characteristics of a DE dynamical system, but the dynamical system it stands for does not need to be a DE one. Furthermore, it is typical to assume that DE SUs communicate with the environment via time-stamped *events*, as opposed to signals. This means that the outputs of SUs can be absent at times where no event is produced.

We adapt the definition of the Discrete Event System Specification(DEVS)[7] in [156] (originally proposed in [10]) to formally define a DE SU S_i, where i denotes the reference of the SU:

$$
\begin{aligned}
& S_i = \left\langle X_i, U_i, Y_i, \delta_i^{ext}, \delta_i^{int}, \lambda_i, ta_i, q_i(0) \right\rangle \\
& \delta_i^{ext} : Q_i \times U_i \rightarrow X_i \\
& \delta_i^{int} : X_i \rightarrow X_i \\
& \lambda_i : X_i \rightarrow Y_i \cup \{NaN\} \\
& ta_i : X_i \rightarrow \mathbb{R}_{\geq 0} \cup \infty \\
& q_i(0) \in Q_i \\
& Q_i = \{(x, e) | x \in X_i \text{ and } 0 \leq e \leq ta_i(x)\}
\end{aligned}
\tag{1}
$$

where:

- X_i, U_i, and Y_i are the set of possible discrete states, input events, and output events, respectively;
- $\delta_i^{ext}(q_i, u_i) = x_i'$ is the external transition function that computes a new total state $(x_i', 0) \in Q_i$ based on the current total state q_i and an input event u_i;
- $\delta_i^{int}(x_i) = x_i'$ is the internal transition function that computes a new total state $(x_i', 0) \in Q_i$ when the current total state is $(x_i, ta_i(x_i)) \in Q_i$;

[7] In the original DEVS definition, the initial state and the absent value in the output function are left implicit. Here we make them explicit, to be consistent with Section 4. Note also that there are many other variants of DE formalisms. For instance, DE in hardware description languages (VHDL and Verilog) and actor based systems (for instance the DE director in Ptolemy II [7]).

- e denotes the elapsed units of time since the last transition (internal or external);
- $\lambda_i(x_i) = y_i \in Y_i \cup \{NaN\}$ is the output event function, invoked right before an internal transition takes place and NaN encodes an absent value;
- $ta_i(x_i) \in \mathbb{R}$ is the time advance function that indicates how much time passes until the next state change occurs, assuming that no external events arrive;
- $q_i(0)$ is the initial state;

The execution of a DE SU is described informally as follows. Suppose that the SU is at time $t_i \in \mathbb{R}_{\geq 0}$ and marks the current discrete state as x_i for $e \geq 0$ elapsed units of time. Since $e \leq ta_i(x_i)$, the total state is $(x_i, e) \in Q_i$. Let $tn = t_i + ta_i(x_i) - e$. If no input event happens until tn, then at time tn an output event is computed as $y_i := \lambda_i(x_i)$ and the new discrete state x_i is computed as $x_i := (\delta_i^{int}(x_i), 0)$. If, on the other hand, there is an event at time $ts < tn$, that is, u_i is not absent at that time, then the solver changes to state $x_i := (\delta_i^{ext}((x_i, e + ts - t_i), u_i), 0)$ instead.

In the above description, if two events happen at the same time, both are processed before the simulated time progresses. Due to the transiency and reactivity properties, the state and output trajectories of a DE SU can only be well identified if the time base, traditionally the positive real numbers, includes a way to order simultaneous events, and simultaneous state changes. An example of such a time base is the notion of superdense time [111][113][43], where each time point is a pair $(t, n) \in \mathcal{T} \times \mathcal{N}$, with $\mathcal{T}$ typically being the positive real numbers and $\mathcal{N}$, called the index, is the set of natural numbers. In this time base, a state trajectory is a function $x_i : \mathcal{T} \times \mathcal{N} \to V_{x_i}$, where V_{x_i} is the set of values for the state, and an output/input trajectory is $u_i : \mathcal{T} \times \mathcal{N} \to V_{u_i} \cup \{NaN\}$. Simultaneous states and events can be formally represented with incrementing indexes. See [80] for an introduction.

Eqs. (2) and (3) show instances of SUs represented in the adapted definition of DEVS.

Algorithm 1 shows a trivial orchestrator[8], which computes the behavior trace of a single DE SU, as specified in Eq. (1), that has no inputs. Remarks: tl holds the time of the last transition; and the initial elapsed time satisfies $0 \leq e \leq ta_i(x_i(0))$;

Algorithm 1: Single autonomous DE SU orchestration.

Data: A $S_i = \langle X_i, \emptyset, Y_i, \delta_i^{ext}, \delta_i^{int}, \lambda_i, ta_i, (x_i(0), e_i) \rangle$.

```
t_i := 0 ;
x_i := x_i(0) ;                          // Initial discrete state
tl := −e_i ;                             // Account for initial elapsed time
while true do
    t_i := tl + ta_i(x_i) ;              // Compute time of the next transition
    y_i := λ_i(x_i) ;                    // Output
    x_i := δ_i^int(x_i) ;                // Take internal transition
    tl := t_i ;
end
```

3.2 DE Co-simulation Orchestration

DE co-simulation scenarios are comprised of multiple DE SUs (Eq. (1)) coupled through output to input connections, which map output events of one SU to external events in other SU.

[8] Algorithm 1 is based on [156] and is originally proposed in [10].

Consider the following DE SUs of a traffic light and a police office, respectively:

$$
\begin{aligned}
&S_1 = \left\langle X_1, U_1, Y_1, \delta_1^{ext}, \delta_1^{int}, \lambda_1, ta_1, q_1(0) \right\rangle \\
&X_1 = Y_1 = \{red, yellow, green, off\} \\
&U_1 = \{toAuto, toOff\}; \qquad q_1(0) = (red, 0) \\
&\delta_1^{ext}((x_1, e), u_1) = \begin{cases} off & \text{if } u_1 = toOff \\ red & \text{if } u_1 = toAuto \text{ and } x_1 = off \end{cases} \\
&\delta_1^{int}(x_1) = \begin{cases} green & \text{if } x_1 = red \\ yellow & \text{if } x_1 = green \\ red & \text{if } x_1 = yellow \end{cases} \\
&\lambda_1(x_1) = \begin{cases} green & \text{if } x_1 = red \\ yellow & \text{if } x_1 = green \\ red & \text{if } x_1 = yellow \end{cases} \\
&ta_1(x_1) = \begin{cases} 60 & \text{if } x_1 = red \\ 50 & \text{if } x_1 = green \\ 10 & \text{if } x_1 = yellow \\ \infty & \text{if } x_1 = off \end{cases}
\end{aligned}
\tag{2}
$$

$$
\begin{aligned}
&S_2 = \left\langle X_2, U_2, Y_2, \delta_2^{ext}, \delta_2^{int}, \lambda_2, ta_2, q_2(0) \right\rangle \\
&X_2 = \{working, idle\} \\
&U_2 = \emptyset \\
&Y_2 = \{toWork, toIdle\} \\
&\delta_2^{int}(x_2) = \begin{cases} idle & \text{if } x_2 = working \\ working & \text{if } x_2 = idle \end{cases} \\
&\lambda_2(x_2) = \begin{cases} toIdle & \text{if } x_2 = working \\ toWork & \text{if } x_2 = idle \end{cases} \\
&ta_2(x_2) = \begin{cases} 200 & \text{if } x_2 = working \\ 100 & \text{if } x_2 = idle \end{cases} \\
&q_2(0) = (idle, 0)
\end{aligned}
\tag{3}
$$

To model a scenario where the police officer interacts with a traffic light, the output events Y_2 have to be mapped into the external events of the traffic light SU. In this example, if $U_1 = \{toAuto, toOff\}$ are the external input events handled by the traffic light SU, the mapping $Z_{2,1} : Y_2 \rightarrow U_1$ is defined by:

$$
Z_{2,1}(y_2) = \begin{cases} toAuto & \text{if } y_2 = toIdle \\ toOff & \text{if } y_2 = toWork \end{cases}
\tag{4}
$$

Based on the idea of abstract SUs [11], we formalize a DE co-simulation scenario with reference cs as follows:

$$
\left\langle U_{cs}, Y_{cs}, D, \{S_d : d \in D\}, \{I_d : d \in D \cup \{cs\}\}, \left\{Z_{i,d} : d \in D \wedge i \in I_d\right\}, \text{Select} \right\rangle
\tag{5}
$$

where:

- U_{cs} is the set of possible input events, external to the scenario;
- Y_{cs} is the set of possible output events from the scenario to the environment;
- D is an ordered set of SU references;
- For each $d \in D$, S_d denotes a DE SU, as defined in Eq. (1);
- For each $d \in D \cup \{cs\}$, $I_d \subseteq (D \setminus \{d\}) \cup \{cs\}$ is the set of SUs that can influence S_d, possibly including the environment external to the scenario (cs), but excluding itself;
- For each $i \in I_d$, $Z_{i,d}$ specifies the mapping of events:

$$
\begin{aligned}
&Z_{i,d} : U_i \rightarrow U_d, \text{ if } i = cs \\
&Z_{i,d} : Y_i \rightarrow Y_d, \text{ if } d = cs \\
&Z_{i,d} : Y_i \rightarrow U_d, \text{ if } i \neq cs \text{ and } d \neq cs
\end{aligned}
$$

- Select : $2^D \rightarrow D$ is used to deterministically select one SU among multiple SUs ready to produce output events simultaneously, i.e., when at time t, the set of SUs

$$IMM(t) = \{d | d \in D \wedge q_d(t) = (x_d, ta_d(x_d))\} \tag{6}$$

has more than one SU reference. In addition, $\text{Select}(IMM(t)) \in IMM(t)$.

The following co-simulation scenario cs couples the traffic light SU to the police officer SU:

$$\begin{gathered} \langle \emptyset, Y_{cs}, \{1,2\}, \{S_1, S_2\}, \{I_1, I_2, I_{cs}\}, \{Z_{2,1}, Z_{1,cs}\}, \text{Select} \rangle \\ Y_{cs} = Y_1; \qquad I_1 = \{2\}; \qquad I_2 = \emptyset; \qquad I_{cs} = \{1\}; \qquad Z_{1,cs}(y_1) = y_1 \end{gathered} \tag{7}$$

where: S_1 is the traffic light SU and S_2 the police officer SU (Eq. (3)); Y_1 is the output of S_1; $Z_{2,1}$ is defined in Eq. (4); and the omitted $Z_{i,d}$ functions map anything to absent (NaN).

The Select function is particularly important to ensure that the co-simulation trace is unique. For example, consider the co-simulation scenario of Eq. (7), and suppose that at time tn both SUs are ready to output an event and perform an internal transition. Should the traffic light output the event and perform the internal transition first, or should it be the police office to do it first? In general, the order in which these output/transition actions are performed matters.

Algorithm 2 illustrates the orchestrator of an autonomous (without inputs) DE co-simulation scenario [9]. Remarks: t_{cs} holds the most recent time of the last transition in the scenario; e_d is the elapsed time of the current state $q_d = (x_d, e_d)$ of S_d; tn is the time of the next transition in the scenario; i^* denotes the chosen imminent SU; I_{cs} is the set of SUs that can produce output events to the environment; y_{cs} is the output event signal of the scenario to the environment; and $\{d | d \in D \wedge i^* \in I_d\}$ holds the SUs that S_{i^*} can influence.

Fig. 2 shows the behavior trace of the traffic light in the co-simulation scenario of Eq. (7).

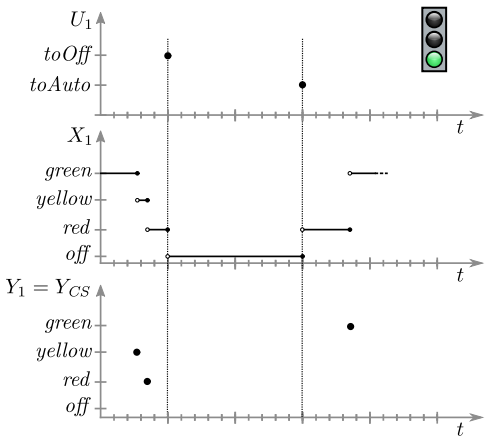


Fig. 2. Example trace of the traffic light.

Algorithm 2 is similar to Algorithm 1: i) The time advance of the scenario ta_{cs} corresponds to the time advance of a single SU; ii) The output produced by the state transition is analogous to the λ function of a single SU; and iii) The output and state transition of child S_{i^*}, together with the external transitions of the SUs influenced by S_{i^*}, are analogous to the internal transition of a single SU. It is natural then that a co-simulation scenario cs as specified in Eq. (5), can be made to behave as a single DE SU S_{cs}. Intuitively, the state of S_{cs} is the set product of the total states of each child DE SU; ta_{cs} is the minimum time until one of the DE SUs executes an internal transition; the internal transition of S_{cs} gets the output event of the imminent SU, executes the external transitions of all the affected SUs, updates the elapsed time of all unaffected SUs, and computes the next state of the imminent SU; the external transition of S_{cs} gets an event from the environment, executes the external transition of all the affected SUs, and updates the elapsed time of all the unaffected SUs [11]. In [152], the formal construction of S_{cs} is provided.

The resulting co-SU S_{cs} behaves exactly as a DE SU specified in Eq. (1). It can thus be executed with Algorithm 1 (in case of no inputs), or composed with other SUs in hierarchical

[9] Algorithm 2 is based on [156]

Algorithm 2: Autonomous DE co-simulation scenario orchestration.

Data: A co-simulation scenario $cs = \langle \emptyset, Y_{cs}, D, \{S_d\}, \{I_d\}, \{Z_{i,d}\}, \text{Select} \rangle$.

$t_{cs} := 0$;

$x_i := x_i(0)$ for all $i \in D$; // Store initial discrete state for each unit

while *true* **do**

  $ta_{cs} := \min_{d \in D} \{ta_d(x_d) - e_d\}$; // Time until the next internal transition

  $tn := t_{cs} + ta_{cs}$; // Time of the next internal transition

  $i^* := \text{Select}(IMM(tn))$; // Get next unit to execute

  $y_{i^*} := \lambda_{i^*}(x_{i^*})$;

  $x_{i^*} := \delta^{int}_{i^*}(x_{i^*})$; // Store new discrete state

  $e_{i^*} := 0$; // Reset elapsed time for the executed unit

  if $i^* \in I_{cs}$ **then**

    $y_{cs} := Z_{i^*,cs}(y_{i^*})$; // Compute output of the scenario

  end

  for $d \in \{d | d \in D \wedge i^* \in I_d\}$ **do**

    $u_d := Z_{i^*,d}(y_{i^*})$; // Trigger internal units that are influenced by unit i^*

    $x_d := \delta^{ext}_d((x_d, e_d + ta_{cs}), u_d)$;

    $e_d := 0$;

  end

  for $d \in \{d | d \in D \wedge i^* \notin I_d\}$ **do**

    $e_d := e_d + ta_{cs}$; // Update the elapsed time of the remaining units

  end

  $t_{cs} := tn$; // Advance time

end

co-simulation scenarios. Hierarchical co-simulation scenarios can elegantly correspond to real hierarchical systems, a natural way to deal with their complexity [5].

In summary, DE based co-simulation exhibits the following characteristics:

reactivity: A DE SU (analogously, a DE co-SU) has to process an event at the moment it occurs.

transiency: In both Algorithm 2 and in a DE co-SU, the time advance ta_{cs} to the next imminent child internal transition can be zero for successive iterations, so an orchestrator has to be able to tolerate the fact that simulated time may not advance for several iterations.

predictable step sizes: In a DE co-simulation scenario without inputs, the orchestrator, as shown in Algorithm 2, can always predict the next simulated time step. In a scenario with inputs, if the environment provides the time of the next event, then the next simulated time step can be predicted too. For this to be possible, black box DE SUs have to be able to inform the orchestrator what their time advance is. This is not a trivial task for DE SUs that simulate continuous systems whose future behavior trace, especially when reacting to future inputs, is not easily predicted without actually computing it.

In the next subsection the main challenges in DE based co-simulation, and the requirements (or capabilities) their solutions impose in DE SUs, are made explicit.

3.3 Challenges

Causality. For the sake of simplicity, Algorithm 2 is sequential. In a hierarchical co-SU, the imminent SU (closest to performing an internal transition) will be the one to execute, thus inducing that there is a global order in the events that are exchanged. This global order avoids causality violations but is too pessimistic. Not every event $y_2(t_2)$ occurring after some event $y_1(t_1)$ has been caused necessarily by $y_1(t_1)$. Moreover, the co-simulation scenario holds information —the dependencies $\{I_d\}$— that can be used to determine who influences what [41][27].

A parallel optimistic orchestrator that takes $\{I_d\}$ into account is, in general, faster in the wall clock time sense, than a pessimistic, sequential one. However, most of these, the Time-warp algorithm [33] being a well known example, require rollback capabilities of SUs. Moreover, in parallel optimistic DE co-simulation, any of the SUs in the scenario needs (theoretically) to support multiple rollbacks and have enough memory to do so for an arbitrary distant point in the past [95].

We make a distinction between multiple rollback and single rollback capabilities. To support single rollback, a SU needs to store only the last committed state, thereby saving memory.

Causality is a compositionality property: if each child SU does not violate causality, then any orchestrator has to ensure that the causality is not violated when these SUs are coupled.

Determinism and Confluence. Determinism is also a compositional property. The Select function, in the co-simulation scenario definition of Eq. (5), is paramount to ensure the compositionality of deterministic behavior. The alternative to the Select function is to ensure that all possible interleavings of executions always lead to the same behavior trace – this is known as *confluence*. Intuitively, if a co-SU is compositional with respect to confluence, then it is also compositional with respect to determinism.

Proving confluence is hard in general for black box DE co-simulation because it depends on knowledge about how the child SUs react to external events, which is potentially valuable IP. Parallel-DEVS [87] is an approach, which leaves the confluence property to be satisfied by the modeler.

Dynamic Structure. Until now, the dependencies $\{I_d\}$, in Eq. (5), have been assumed to be fixed over time. From a performance perspective, a static sequence of dependencies may be too conservative, especially if used to ensure causality in optimistic parallel co-simulation. If, throughout a parallel co-simulation, a SU S_1 seldom outputs events that influence S_2, it makes sense that most of the time t, $S_1 \notin I_2$. Dynamic structure co-simulation allows for $\{I_d\}$ to change over time, depending on the behavior trace of the SUs. It can be used to study self-organizing systems [135][17].

Distribution. Co-SUs whose child SUs are geographically distributed are common [95]. Interesting solutions like computation allocation [119][138], bridging the hierarchical encapsulation [139], and the use of dead-reckoning models [42] have been proposed to mitigate the additional communication cost. Moreover, security becomes important, as pointed out, and addressed, in [121].

4 CONTINUOUS-TIME-BASED CO-SIMULATION

In the continuous time (CT) based co-simulation approach, the orchestrators' and simulation units' (SUs) behavior and assumptions are borrowed from the CT system simulation domain. We describe these below.

4.1 CT Simulation Units

A CT SU is assumed to have a state that evolves continuously over time. It is easier to get the intuitive idea of this by considering a SU of a CT dynamical system, such as a mass-spring-damper, depicted in the left hand side of Fig. 3. The state is given by the displacement x_1 and velocity v_1 of the mass, and the evolution by:

$$\begin{aligned} \dot{x_1} &= v_1; & m_1 \cdot \dot{v_1} &= -c_1 \cdot x_1 - d_1 \cdot v_1 + F_e \\ x_1(0) &= p_1; & v_1(0) &= s_1 \end{aligned} \tag{8}$$

where $\dot{x}$ denotes the time derivative of x; c_1 is the spring stiffness constant and d_1 the damping coefficient; m_1 is the mass; and F_e denotes an external input force acting on the mass over time. Fig. 1b shows an example of a behavior trace.

Eq. (8) can be generalized to the state space form:

$$\dot{x} = f(x,u) \quad ; \quad y = g(x,u) \quad ; \quad x(0) = x_0 \tag{9}$$

where x is the state vector, u the input and y the output vectors, and x_0 is the initial state. If $f(x,u)$ is sufficiently differentiable, x can be approximated with a truncated Taylor series [58][1]:

$$x(t+h) = x(t) + f(x(t),u(t)) \cdot h + O\left(h^2\right) \tag{10}$$

where $h \geq 0$ is the micro-step size. Eq. (10) is the basis of a family of numerical solvers that iteratively compute an approximated behavior trace $\tilde{x}$.

A CT SU is assumed to have a behavior that is similar to one of a numerical solver computing a set of differential equations. We reinforce that this does not restrict CT SUs to being mockups of CT systems, even though it is easier to introduce them as such. For example, a SU S_1 that simulates the mass-spring-damper system takes as input the external force $F_e(t)$, applies Eq. (10) to Eq. (8), to compute the new state $[x(t+h), v(t+h)]^T$, and outputs it.

4.2 CT Co-simulation Orchestration

Consider now a SU S_2 for the system depicted in the right hand side of Fig. 3. It takes the displacement x_c of the left end of the spring/damper and its derivative $\dot{x_c}$, and outputs the reaction force F_c. Suppose S_1 and S_2 are coupled, setting $x_c = x_1$, $\dot{x_c} = v_1$ and $F_e = F_c$, so that the resulting co-simulation scenario represents the multi-body system in Fig. 3.

In CT based co-simulation, to overcome the fact that each SU's micro-step sizes are independent, a communication step size H (also known as macro-step size or communication grid size) has to be defined. H marks the times at which the SUs exchange values of inputs/outputs.

Suppose a SU S_i is at time $n \cdot H$, for some natural n, and is asked by an orchestrator to execute until time $(n+1) \cdot H$. If S_i only gets its inputs valued at $n \cdot H$, then extrapolation must be used to get the inputs in any of the internal micro-steps of the SU. In other words, when time is $n \cdot H + m \cdot h_i$, for $m \leq \frac{H}{h_i}$ and micro-step size h_i, an extrapolation function $\phi_{u_i}(m \cdot h_i, u_i(n \cdot H), u_i((n-1) \cdot H), \ldots)$, built from input values known at previous communication time points, is used to approximate the value of $u_i(n \cdot H + m \cdot h_i)$. Analogously, interpolation techniques have to be used

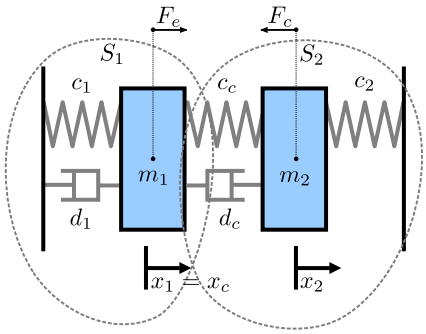


Fig. 3. A multi-body system comprised of two mass-spring-damper subsystems.

when the orchestrator makes the input value available at time $(n+1)\cdot H$ but the SU is still at time $n\cdot H$. In the simplest case, the extrapolations can be constant. In the coupled mass-spring-dampers, this means:

$$\phi_{F_e}(t, F_e(n\cdot H)) = F_e(n\cdot H); \quad \phi_{x_c}(t, x_c(n\cdot H)) = x_c(n\cdot H); \quad \phi_{\dot{x}_c}(t, \dot{x}_c(n\cdot H)) = \dot{x}_c(n\cdot H) \tag{11}$$

In the state of the art, input extrapolation approaches can be classified as: Constant; Linear; Polynomial; Extrapolated-Interpolation [90][23]; Context-aware [70][19]; and Estimated Dead-Reckoning Model [129][78][57]; See [23][14][55][147] for an overview of linear and higher order extrapolation techniques and how these affect the accuracy of the co-simulation trace.

We are now ready to formally define the behavior of a CT SU S_i:

$$\begin{aligned} &S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle \\ &\delta_i : \mathbb{R}\times X_i \times U_i \rightarrow X_i \\ &\lambda_i : \mathbb{R}\times X_i \times U_i \rightarrow Y_i \text{ or } \mathbb{R}\times X_i \rightarrow Y_i \\ &x_i(0) \in X_i \\ &\phi_{U_i} : \mathbb{R}\times U_i \times \ldots \times U_i \rightarrow U_i \end{aligned} \tag{12}$$

where:

- X_i is the state vector space;
- U_i is the input vector space;
- Y_i is the output vector space;
- $\delta_i(t, x_i(t), u_i(t)) = x_i(t+H)$ or $\delta_i(t, x_i(t), u_i(t+H)) = x_i(t+H)$ is the function that instructs the SU to compute a behavior trace from t to $t+H$, making use of the input extrapolation (or interpolation) function ϕ_{U_i};
- $\lambda_i(t, x_i(t), u_i(t)) = y_i(t)$ or $\lambda_i(t, x_i(t)) = y_i(t)$ is the output function; and
- $x_i(0)$ is the initial state.

A CT co-simulation scenario with reference cs includes at least the following information[10]:

$$\begin{aligned} &\langle U_{cs}, Y_{cs}, D, \{S_i : i \in D\}, L, \phi_{U_{cs}} \rangle \\ &L : (\Pi_{i\in D} Y_i) \times Y_{cs} \times (\Pi_{i\in D} U_i) \times U_{cs} \rightarrow \mathbb{R}^m \end{aligned} \tag{13}$$

where D is an ordered set of SU references, each S_i is defined as in Eq. (12), $m \in \mathbb{N}$, U_{cs} is the vector space of inputs external to the scenario, Y_{cs} is the vector space of outputs of the scenario, $\phi_{U_{cs}}$ a set of input approximation functions, and L induces the SU coupling constraints, that is, if $D = \{1, \ldots, n\}$, then the coupling is the solution to $L(y_1, \ldots, y_n, y_{cs}, u_1, \ldots, u_n, u_{cs}) = \bar{0}$, where $\bar{0}$ denotes the null vector. As an example, the co-simulation scenario representing the system of Fig. 3 is:

$$cs = \langle \emptyset, \emptyset, \{1,2\}, \{S_1, S_2\}, L, \emptyset \rangle ; \qquad L = [x_c - v_1;\ \dot{x}_c - x_1;\ F_e - F_c]^T \tag{14}$$

Algorithm 3 summarizes, in a generic way, the tasks of the orchestrator related to computing the co-simulation of a scenario cs with no external inputs. It represents the Jacobi communication approach: SUs exchange values at time t and independently compute the trace until the next communication time $t+H$. The way the system in Eq. (15) is solved depends on the definition of L. In the most trivial case, the system reduces to an assignment

[10] Please note that this formalization is related to the formalization proposed by [79], with the main differences: i) it is not designed to formalize a subset of the FMI Standard, ii) it accommodates algebraic coupling conditions, and iii) it does not explicitly define port variables.

of an output $y_j(t)$ to each input $u_i(t)$, and so the orchestrator just gets the output of each SU and copies it onto the input of some other SU, in an appropriate order. Concrete examples of Algorithm 3 are described in [69][151][107][96][91][104][83][64].

An alternative to the Jacobi communication approach is the Gauss-Seidel (a.k.a. sequential or zig-zag) approach, where an order of the SUs' δ function is forced to ensure that, at time t, they get inputs from a SU that is already at time $t + H$. Gauss-Seidel approach allows for interpolations of inputs, which is more accurate, but hinders the parallelization potential. Examples are described in [15][69][14][136].

Algorithm 3: Generic Jacobi based orchestrator for autonomous CT co-simulation scenarios.

Data: An autonomous scenario $cs = \langle \emptyset, Y_{cs}, D = \{1, \ldots, n\}, \{S_i\}, L, \emptyset \rangle$ and a communication step size H.

Result: A co-simulation trace.

$t := 0$;

$x_i := x_i(0)$ for $i = 1, \ldots, n$;

while *true* **do**

Solve the following system for the unknowns:

$$\begin{cases} y_1 = \lambda_1(t, x_1, u_1) \\ \ldots \\ y_n = \lambda_n(t, x_n, u_n) \\ L(y_1, \ldots, y_n, y_{cs}, u_1, \ldots, u_n) = \bar{0} \end{cases} \tag{15}$$

$x_i := \delta_i(t, x_i, u_i)$, for $i = 1, \ldots, n$; // Instruct each SU to advance

$t := t + H$; // Advance time

end

Similarly to DE based co-simulation, a CT co-simulation scenario, together with an orchestrator, should behave as a (co-)SU of the form of Eq. (12), and thus be coupled with other SUs, forming hierarchical co-simulation scenarios: the state of the co-SU is the set product of the states of the internal SUs; the inputs are given by U_{cs} and the outputs by Y_{cs}; the transition and output functions are implemented by the orchestrator; the communication step size H used by the orchestrator is analogous to a SU's micro-step sizes, and the input extrapolation function is ϕ_{U_i}.

Algorithm 3 makes it clear that the SUs can be coupled with very limited information about their internal details. However, the *blind* coupling can lead to compositionality problems, as will be discussed in the sections below. The common trait in addressing these is to require more from the individual SUs: either more capabilities, or more information about the internal (hidden) dynamical system.

4.3 Challenges

Modular Composition – Algebraic Constraints. In the co-simulation scenario described in Eq. (14), the coupling condition L translates into a set of assignments from outputs to inputs. In practice, the SUs' models are not created with a specific coupling pattern in mind and L can be more complex. As an example, adapted from [53], consider the system coupled by a massless rigid link, depicted in Fig. 4. The input to S_3 is the coupling force F_c, and the output is the state of the mass $[\tilde{x}_3, \tilde{v}_3]^T$. The input to S_1 is the external force F_e and

the outputs are the state of the mass $[\tilde{x}_1, \tilde{v}_1]^T$. There is clearly a mismatch. However, the massless link restricts the states and inputs of the two SUs to be the same. Whatever the input forces may be, they are equal and opposite in sign. Hence, any orchestration algorithm has to find inputs that ensure the coupling constraints are satisfied:

$$L = [\tilde{x}_1(n \cdot H) - \tilde{x}_3(n \cdot H); \quad \tilde{v}_1(n \cdot H) - \tilde{v}_3(n \cdot H); \quad F_e(n \cdot H) + F_c(n \cdot H)]^T = \bar{0} \tag{16}$$

This problem has been addressed in [104][31][15][14][55][53][56]. A common feature of the solutions proposed is to require that each SU provides the sensitivity of its outputs with respect to its inputs, and be able to rollback to previous states.

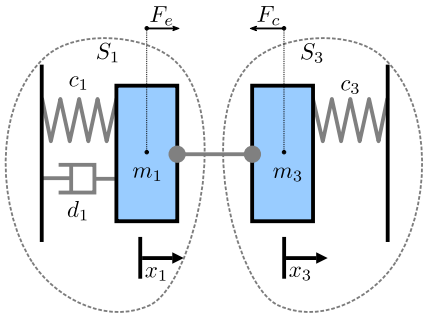


Fig. 4. A multi-body system coupled by a massless link.

To understand why the black box nature of SUs affects their modularity, note that, if the equations of both constituent systems in the example are made available and coupled (a white box approach), they can be simplified to a lumped mass-spring-damper, which is easily solvable. Such an approach is common in acausal modeling languages, such as Modelica [157]. In the white-box approach, the same constituent system can be coupled to other systems in many different contexts, whereas in co-simulation it is possible to get around the modularity aspect, but at a cost.

Algebraic loops. Algebraic loops occur whenever there is a variable that indirectly depends on itself. We distinguish two kinds of algebraic loops in co-simulation [40]: the ones spanning just input variables, and the ones that include state variables as well. The first kind arises when the outputs of a SU depend on its inputs, while the second kind happens when implicit numerical solvers are used, or when the input approximating functions are interpolations. As shown in [40][66] (and empirically in [69]), neglecting a loop can lead to a prohibitively high error in the co-simulation. Instead, fixed point iteration technique should be used to solve algebraic loops. For those involving state variables, the same co-simulation step has to be repeated until convergence, whereas for loops over inputs/outputs, the iteration just repeats the evaluation of the output functions.

An orchestrator that makes use of rollback to repeat the co-simulation step with corrected inputs is called dynamic iteration, waveform iteration, and strong or onion coupling [153][133]. If the SUs expose their outputs at every internal micro-step, then the waveform iteration can be used [112]. Strong coupling approaches are typically the best in terms of accuracy, but worst in terms of performance. A variant that attempts to obtain the middle-ground is the so-called semi-implicit method, where a fixed limited number of correction steps is performed. See [55][53] for examples of this approach.

In the current FMI Standard for co-simulation, it is *not* possible, in the *step mode*, to perform a fixed point iteration on the output variables only. A workaround is to use a strong coupling technique.

Until here, we have assumed full knowledge of the models being simulated in each SU to explain how to identify, and deal with, algebraic loops. In practice, with general black-box SUs, extra information is required to identify algebraic loops. According to [79][66][20], a binary flag denoting whether an output depends directly on an input is sufficient. A

structural analysis, for example, with Tarjan's strong component algorithm [131], can then be performed to identify the loops.

Consistent Initialization of Simulators. The definition of a SU in Eq. (12) assumes that an initial condition is part of the SU. However, as seen in the example of Fig. 4, the initial states of the SUs can be coupled by algebraic constraints, through the output functions, which implies that the initial states of the SUs cannot be set independently of the co-simulation in which they are used. In general, for a co-simulation scenario as defined in Eq. (13), there is an extra coupling function L_0 that at the time $t = 0$, has to be satisfied. For example:

$$L_0(x_1(0), \ldots, x_n(0), y_1(0), \ldots, y_n(0), y_{cs}(0), u_1(0), \ldots, u_n(0), u_{cs}(0)) = \bar{0} \tag{17}$$

Eq. (17) may have an infinite number of solutions or have algebraic loops. The initialization problem (or co-initialization) is identified in [72] and addressed in [96]. In the FMI Standard, there is a dedicated mode for the (possibly fixed point iteration based) search of a consistent initial state in all SUs.

Compositional Convergence – Error Control. In the context of co-simulation of CT systems, the most accurate trace is the analytical solution to the coupled model that underlies the co-simulation scenario. In practice, the analytical solution for a coupled model cannot be found easily. Calculating the error precisely is therefore impossible for most cases but getting an estimate in how it grows is a well understood procedure in numerical analysis.

In simulation, the factors that influence the error are [1]: the model, the solver, the micro-step size, and, naturally, the size of the time interval to be simulated. In co-simulation, the extrapolation functions introduce error in the inputs of the SUs, which is translated into error in the state/outputs of these, causing a feedback on the error that can increase over time. Intuitively, the larger the co-simulation step size H, the larger is the error made by the extrapolation functions.

For a solver to be useful, it must be convergent, that is, the computed trace must coincide with the accurate trace when $h \to 0$ [9]. It means the error can be controlled by adjusting the micro step size h. The same concept of convergence applies to co-simulation but does, as the intuition suggests, decreasing the communication step size H lead to a more accurate co-simulation trace? This cannot be answered yet in general co-simulation because the behavior of the coupled model induced by the coupling of SUs may not satisfy Lipschitz continuity.

According to [32][83][40][15][66], if the SUs are convergent and the coupled model induced by the scenario coupling conditions can be written in the state space form of Eq. (9), then the co-SU induced by any of the Jacobi, Gauss-Seidel, or Strong coupling methods, is convergent, regardless of the polynomial extrapolation technique used. Presence of algebraic loops, or complex coupling constraints, are factors that may make it impossible to write the coupled model in state space form [14].

For a convergent co-SU, some of the techniques traditionally used in simulation, have been applied in co-simulation to *estimate* the error during the computation: Richardson extrapolation [96][66][16]; Multi-Order Input Extrapolation [24][83]; Milne's Device [14][15][54][53][55]; Parallel Embedded Method [153]; and Conservation Laws [51].

After the error is deemed too large by one of the above methods, the correction can be applied pessimistically (rolling back and repeating the same step) or optimistically (adapt the next step). To mitigate the overhead of a pessimistic approach, the corrections may be applied only to sensitive SUs, as carried out in [63].

Compositional Stability. Contrarily to convergence, numerical stability is a property that depends on the characteristics of the system being co-simulated. One of the ways numerical stability in co-simulation can be studied is by calculating the spectral radius of the error in the co-SU, written as an autonomous linear discrete system [82]. See [152] for an example.

Different coupling methods, and different approximation functions yield different stability properties. See [23][84][82] for the stability analysis of multiple coupling approaches and approximating functions. Stability of various co-SUs has been also studied in [39][54][36][104][14]. The *rules of thumb* drawn from these papers can be summarized as: (1) Co-simulators that employ fixed point iteration techniques typically have better stability properties; (2) Gauss-Seidel coupling approach has slightly better stability properties when the order in which the SUs compute is appropriate (e.g., the SU with the highest mass should be computed first [14]).

Compositional Continuity. If a SU is a mock-up of a CT system, then it is reasonable to expect that its inputs are also continuous. As discussed in [51][23], the careless use of input extrapolations (e.g., constant extrapolation) may violate this assumption.

Any *sudden* change in the input to a CT SU may wreak havoc in the performance of its simulator, causing it to reduce inappropriately the internal micro step size, to reinitialize the solver [1], to discard useful information about the past (in multi-step solvers [147][148]), and/or produce inaccurate values in its input extrapolation [49]. Furthermore, a discontinuity may be propagated to other SUs, aggravating the problem.

A solution to avoid discontinuities in the input approximations is to use the extrapolated interpolation methods [23][90].

Real-time Constraints, Noise, and Delay. As introduced in Section 2, the major challenge in real-time simulation is to ensure that a SU is fast-enough to satisfy the timing constraint $t = \alpha\tau$. In real-time co-simulation, this challenge gets aggravated due to the presence of multiple SUs, with different capabilities [130], and whose internal workings are unknown. Furthermore, real-time co-simulation is often used when at least one of the SUs is a physical entity. This means that measurements may carry noise, and the extrapolation functions used in the other SUs have to be properly protected from that noise (e.g., using statistical techniques such as Kalman filtering [35][57]). Finally, the quality of the network is important, as the real-time SUs needs to receive their inputs in a timely manner. To mitigate this, the orchestration algorithm has to compensate for any delays in the receiving of data, and provide inputs to the real-time SU [129].

5 HYBRID CO-SIMULATION APPROACH

Sections 3 and 4 described the essential characteristics and assumptions of simulation units (SUs) for each kind of co-simulation approach. When compared to a CT SU, whose state evolves continuously in time and whose output may have to obey to physical laws of continuity, a DE SU state can assume multiple values at the same time (transiency) and its output is discontinuous. For an orchestrator, a CT SU has some flexibility (safe for algebraic loops and complex coupling conditions) in deciding the parameters (e.g., step size or tolerance) of the co-simulation. In contrast, a DE SU has to get inputs and produce outputs at the precise time an event is supposed to occur, and there is no Lipschitz continuity conditions to help predict how a delay in the output of the DE SU can affect the overall co-simulation trace.

These differences are at the heart of many challenges in hybrid co-simulation scenarios.

5.1 Hybrid Co-simulation Scenarios

We do not give a formal definition of a hybrid co-simulation scenarios because that is related to finding an appropriate standard for hybrid co-simulation, which is a non trivial challenge (see Section 5.2) [80].

Instead, we define it broadly as mixing the characteristics and assumptions of both kinds of SUs. These scenarios, together with an adequate orchestrator, can be used as mock-ups of hybrid systems [26][43][12][25]. A thermostat regulating the temperature in a room is a classical example [155]. The Continuous Time (CT) constituent system represents the temperature dynamics of the room, accounting for a source of heat (radiator). The Discrete Event (DE) part is a controller that turns on/off the radiator depending on the temperature.

The SU S_1 simulates the following dynamics:

$$\dot{x} = -\alpha\,(x - 30q)\,; \qquad x(0) = x_0 \tag{18}$$

where x is the output temperature in the room, $\alpha > 0$ denotes how fast the room can be heated (or cooled) down, and $q \in \{0, 1\}$ is the control input that turns on/off the radiator. The SU S_2 simulates the statemachine shown in Fig. 5, where one can think of the input event *tooHot* as happening when $x(t) \geq 21$ and *tooCold* when $x(t) \leq 19$. The output events *off* and *on* will assign the appropriate value to the input q of S_1. Therefore, the temperature $x(t)$ is kept within a comfort region.

Clearly, the two SUs cannot just be coupled together via input to output assignments. Any orchestrator for this co-simulation scenario has to reconcile the different assumptions about the inputs and output of each SU. The coupling of CT and DE black box SUs has been studied in the state of the art. In essence, two approaches are known, both based on adapting (or wrapping) the behavior of the SU:

Hybrid DE – adapt every CT SU as a DE SU, and use a DE based orchestrator;

Hybrid CT – wrap every DE SU to become a CT SU and use a CT based orchestrator.

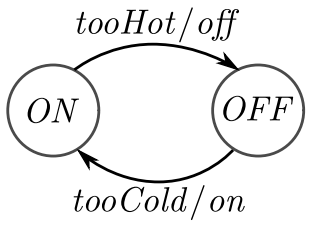


Fig. 5. Statemachine model of the controller constituent system.

According to the formalization that we have proposed for CT and DE SUs, the *Hybrid DE* approach, applied to the thermostat example may involve: adapting S_1 as a DE SU, S_1', with a time advance that matches the size of the co-simulation step; and keeping track of the output of S_1 in order to produce an output event whenever it crosses the thresholds. Conversely, any output event from S_2 has to be converted into a continuous signal for the input $q(t)$ of S_1. Other examples of *Hybrid DE* are described in [140][128][122][48][38][142][74][93][68][143][144][75][86][85][106][120][108].

The *Hybrid CT*, in our example, requires the adaptation of the DE S_2 as a CT SU that takes as input the temperature continuous signal, and internally reacts to an event caused by the crossing of the threshold. The output event of S_2 can be converted into a continuous signal $q(t)$. Examples of the *Hybrid CT* include [97][50][109][89][132][92][134].

Regardless of the approach taken, the properties of the constituent systems have to be retained: the fact that an otherwise discontinuous signal becomes continuous as a result of a linear or higher order extrapolation may not respect the properties of the coupled system. Knowledge of the domain and the SUs is paramount to retain aforementioned properties.

A third option, compared to only using Hybrid CT or Hybrid DE, is to have different mechanisms of orchestrating the SUs depending on the semantic domain. For instance, in the actor modeling language Ptolemy II [7], an actor has many similarities to a SU. Instead

of using either Hybrid CT or Hybrid DE, a so called *Director* block is used for a particular set of connected actors. In this context, the notion of superdense time is fundamental, as discussed in [80] and [29].

In the section below, different issues that arise in hybrid co-simulation will be described. These should be read in the light of hierarchical hybrid co-simulation scenarios, where compositionality is important.

5.2 Challenges

Semantic Adaptation and Model Composition. While a generic wrapper based on the underlying model of computation of the SU can be used, as done in [7][88], the realization of any of the approaches *Hybrid DE* or *Hybrid CT* depends on the concrete co-simulation scenario and the features of the SUs [77][117], as shown with the thermostat example. There is simply no best choice of wrappers for all scenarios. Even at the technical level, the manner in which the events or signals are sent to (or obtained from) the SU may need to be adapted [134]. To account for this variability, the most common adaptations can be captured in a configuration language, as was done in [114][89], or in a specialization of a model of computation, as done in [108][116][125]. These approaches require that a person with the domain knowledge describes how the SUs can be adapted.

Our choice of wrapper for the *Hybrid DE* approach is meant to highlight another problem with the adaptations of SUs: the wrapper incorporates information that will ultimately have to be encoded in the software controller. As such, we argue that the need for sophisticated semantic adaptations should be smaller in later stages of the development of the components so that, for more refined models of the thermostat, the decision about when to turn off the radiator is not made by a wrapper of S_1.

Predictive Step Sizes and Event Location. In the *Hybrid DE* approach, the time advance has to be defined (recall Eq. (1)). Setting it to whatever co-simulation step size H the orchestrator decides will work, but the adapted SU may produce many absent output events. Better adaptations have been proposed. In the thermostat example, S_1' can propose a time advance that coincides with the moment that $x(t)$ will leave the comfort region, thereby always being simulated at the relevant times.

Naturally, these approaches rely on information that may expose the IP of SUs. Others try to adaptively guess the right time advance by monitoring other conditions of interest, set over the own dynamics of the adapted SU, the most common approach being the quantization of the output space [145][75][37][38][123].

The capability to predict the time advance is also useful to enhance the performance/accuracy of CT based co-simulation, as shown in [79].

Locating the exact time at which a continuous signal crosses a threshold (e.g., finding t such that the temperature $x(t) = 19$) is a well known problem [76][146][21] and intimately related to guessing the right time advance for predicting the step size [86][96]. To address this, solutions typically require derivative information of the signal that causes the event, and/or the capability to perform rollbacks.

Discontinuity Identification. In a general hierarchical co-simulation, a SU's output may be an event signal coming from a wrapper of a CT SU, or vice-versa. In any case, at runtime, a signal is often represented as a set of time-stamped points. Observing this sequence of points alone does not make it possible to discern a steep change in a continuous signal, from a true discontinuity, that occurs in an event signal [111][80][146][115]. Extra information is currently used: *a*) a formalization of time which include the notion of absent signal, as

proposed in [132][111][80]; or *b*) an extra signal can be used to discern when a discontinuity occurs, as done in the FMI for Model Exchange [72], even facilitating the location of the exact time of the discontinuity; or *c*) symbolic information (e.g., Dirac impulses [2]) that characterize a discontinuity can be included, as done in [47][103].

Discontinuity Handling. Once a discontinuity is located, how it is handled depends on the nature of the SUs and their capabilities. If the SU is a mock-up of a continuous system then, traditionally, discontinuities in the inputs should be handled by reinitializing the SU [1]. This step can incur a too high performance cost, especially with multi-step numerical methods, and [148][147] proposes an improvement for these solvers. Furthermore, a discontinuity can cause other discontinuities, producing a cascade of re-initializations. During this process, which may not finish, care must be taken to ensure that physically meaningful properties such as energy distribution, are respected [44].

Algebraic Loops, Legitimacy, and Zeno Behavior. Algebraic loops are non-causal dependencies between SUs that can be detected using feedthrough information, as explained in Section 4.3. In CT based co-simulation, the solution to algebraic loops can be attained by a fixed point iteration technique, as covered in Section 4.3. There is the possibility that the solution to an algebraic loop will fail to converge. The result is that, if left unchecked, the orchestrator would move an infinite number of input and output values between SUs, at the same point in time.

In DE based co-simulation a related property is legitimacy [11], which is roughly the undesirable version of the *transiency* property, explained in Section 3. An illegitimate co-simulation scenario will cause the co-simulation orchestrator to move an infinite number of events with the same timestamp between SUs, never advancing time. Distance matrices, used to optimize parallel optimistic approaches, as explained in [95] and used in [98], can be leveraged to detect statically the presence of *some* classes of illegitimacy.

A similar behavior, but more difficult to detect is Zeno behavior. It occurs when there is successively smaller intervals of time between two consecutive events, up to the point that the sum of all these intervals is finite [8]. As shown in [149], a simulator eventually fails to detect the consecutive events. In particular, he advocates that the zeno behavior is a property of the model, whereas the incorrectness is due to a simulation approximation error. However, while illegitimate behaviors are not desired in pure DE co-simulation, Zenoness can be a desired feature in some hybrid co-simulation scenarios (e.g., see [22]). We say in the theoretical sense because, for the purposes of co-simulation, scenarios with Zenoness still have to be recognized and appropriate measures, such as regularization [34], have to be taken.

Stability under X. If a hybrid co-simulation represents a hybrid or switched system [8], then it is possible that a particular sequence of events causes the system to become unstable, even if all the individual continuous modes of operation are stable [4, Example 1.1]. New analyses are required to identify whether the CT SUs can yield unstable trajectories as a result of: (1) noisy inputs; (2) data quantization; (3) change of co-simulation orchestration [102]; (4) the events of wrapped DE SUs [101]; and, (5) delayed exchange of values.

Theory of DE Approximated States. In a pure DE based co-simulation, if round-off errors are neglected, the computed trajectories are essentially exact. To the best of our knowledge, only [11] addresses theoretically how the error in a discrete event system can be propagated. In CT based co-simulation however, error is an accepted and well studied and techniques exist to control it.

In Hybrid co-simulation, there is a need for analysis techniques that provide bounds on the error propagation in the DE SUs, when these are coupled to sources of error.

Standards for Hybrid Co-simulation. While for CT co-simulation there is the Functional Mockup Interface (FMI) standard [72], and for DE co-simulation there is the High Level Architecture (HLA) [158] standard, as of the time of writing, both standards have limitations for hybrid co-simulation. References [132][73][97][29] use/propose extensions to the FMI standard and [67] proposes techniques to perform CT simulation conforming to HLA. Recognizing that hybrid co-simulation is far from well studied, [80] proposes a set of idealized test cases that any hybrid co-SU, and underlying standard, should pass. In particular, it is important to have correct handling and representation of time, to achieve a sound approach for simultaneity.

Finally, even with a standardized interface, SUs have different capabilities: a fact that makes coding an optimal orchestration algorithm difficult. A possible approach to deal with this heterogeneity, proposed in [100], is to assume that all SUs implement the same set of features, code the orchestration algorithm for those features, and delegate to wrappers the responsibility of leveraging extra features (or mitigating the lack of). In the section below, these features are classified.

6 CLASSIFICATION AND APPLICATIONS

Having described the multiple facets of co-simulation, this section summarizes our classification and methodology, and applies it to a typical use case.

6.1 Methodology

To find an initial set of papers related to co-simulation, we used Google Scholar with the keywords “co-simulation”, “cosimulation”, “coupled simulation”, and collected the first 10 pages of papers. Every paper was then filtered by the abstract, read in detail, and its references collected. To guide our reading to the most influential papers, we gave higher priority to most cited (from the papers that we have collected).

We read approximately 30 papers to create the initial version of the taxonomy. Then, as we read new papers, we constantly revised the taxonomy and classified them.

After a while, new references did not cause revisions to the taxonomy, which prompted us to classify the collected papers in a more systematic fashion: all the papers that we collected from 2011 (inclusive) up to, and including, 2016 were classified. Two main reasons justify the last 5 years interval: limited time; and most of the papers refer to, and are based on, prior work. In total, 84 papers were read and classified.

6.2 Taxonomy

The taxonomy is represented as a feature model [154] structured in three main categories, shown in Fig. 6:

Non-Functional Requirements (NFRs): Groups concerns (e.g., performance, accuracy, and IP Protection) that the reference addresses.

Simulation unit (SU) Requirements (SRs): Features required/assumed from the SUs by the orchestrator described in the paper. Examples: Information exposed, causality, local/remote availability, or rollback support.

Framework Requirements (FRs): Features provided by the orchestrator. Examples: dynamic structure, adaptive communication step size, or strong coupling support.

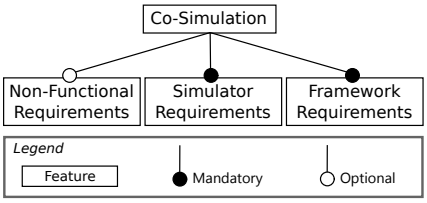


Fig. 6. Top-level.

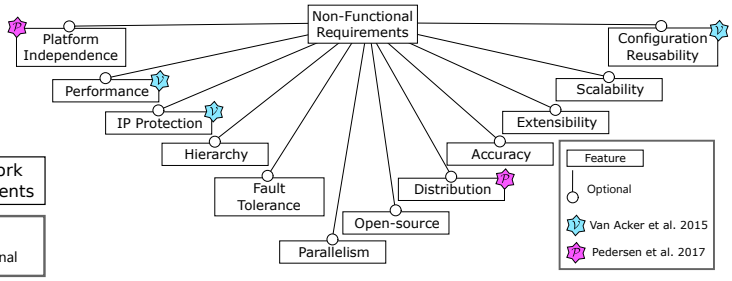


Fig. 7. Non-Functional Requirements.

Each main group is detailed in Figs. 7 to 9. Abstract features denote concepts that can be easily detailed down but we chose not to, for the sake of brevity. Mandatory features are required for the activity of co-simulation, while optional are not.

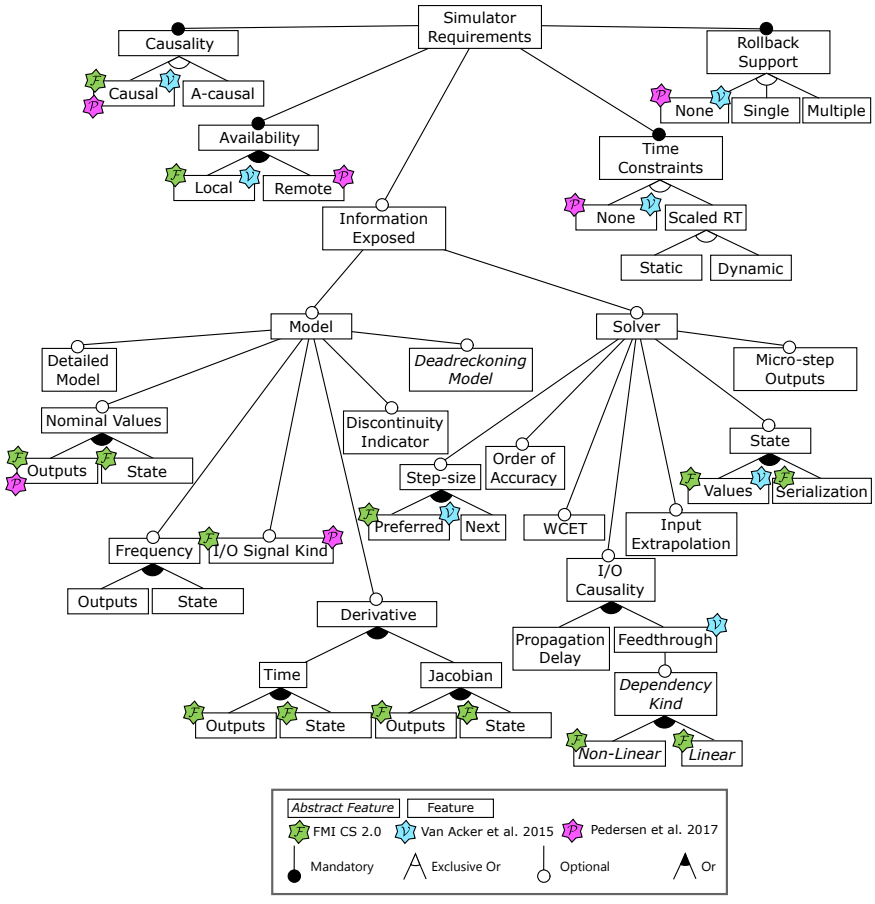


Fig. 8. Simulation Unit Requirements and features provided in the FMI Standard for co-simulation, version 2.0.

6.3 Applications

To demonstrate how the taxonomy is used, we picked three examples from the state of the art: an industrial use case, a co-simulation framework, and a co-simulation standard.

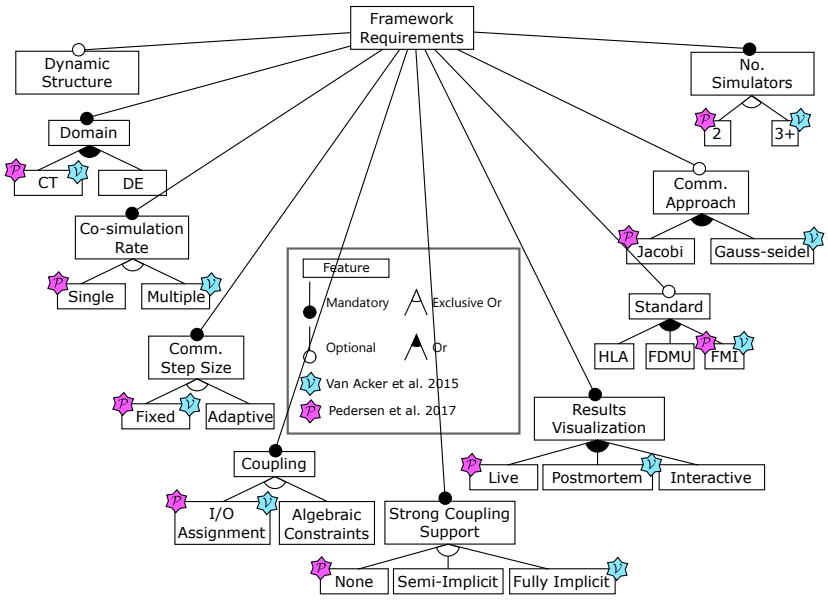


Fig. 9. Framework Requirements.

6.3.1 An Industrial Application. The case study reported in [126] applies co-simulation as part of the development of a controller for an exhaust gas recirculation water handling system. The purpose of this system 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 deposit in a receiving tank. Then, the (dirty) water is pumped to a water treatment center (externally developed) to be purified and reused.

The system is a representative example because: it includes parts that are developed by other departments (e.g., the ship engine) and external suppliers (e.g., the water treatment system); there are both continuous and discrete event dynamics (e.g., the control system is comprised of a state machine and a PI-Controller); and, quoting the authors, "to improve the control strategy of the WHS, a higher-fidelity model [of the systems interacting with the controller] should be used." [126, Section 3.4].

In fact, thanks to the FMI Standard, its support by MATLAB/Simulink®, and to the INTO-CPS co-simulation framework, the authors were able to combine the behavior of higher fidelity models, with the behavior of the controller under development, simulated by an in-house C++ software application framework.

Through co-simulation, it was possible to reproduce and correct an issue that was previously encountered only during a (costly) Hardware-in-the-loop simulation with a physical engine test bench available at the MDT research center in Copenhagen.

This work is classified as highlighted in Figs. 7 to 9.

6.3.2 A Framework. We next consider the work of [136], where an FMI based multi-rate orchestration algorithm is generated from a description of the co-simulation scenario. In the paper, the description language introduced can be reused in a tool-agnostic manner. The orchestration code generator analyzes the co-simulation scenario, and: a) identifies algebraic loops using I/O feedthrough information; b) separates the fast moving SUs from the slow moving ones, using the preferred step size information, and provides interpolation to the fast ones (multi-rate); and c) finds the largest communication step size that divides all step

sizes suggested by SUs and uses it throughout the whole co-simulation. The algebraic loops are solved via successive substitution of inputs, storing and restoring the state of the SUs.

Based on these facts, [136] is classified as highlighted in Figs. 7 to 9.

6.3.3 A Standard. The FMI standard for co-simulation, version 2.0 [150], can also be classified according to the assumptions it makes about the participating SUs. This is highlighted in Fig. 8.

6.4 The State of the Art

The remaining state of the art is classified in Figs. 10–12. The raw data is available online[11] and a more detailed description of each concept is given in [152]. The apparent lack of papers in the interval 2006-2009 is a consequence of our methodology (recall Section 6.1).

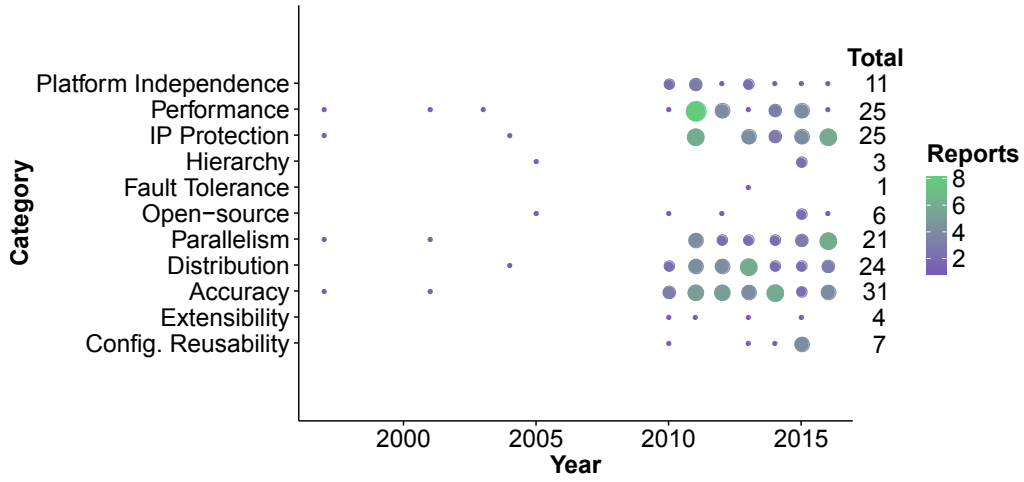


Fig. 10. Classification with respect to non-functional requirements.

6.5 Discussion

Analyzing Fig. 10, Accuracy is the most observed NFR, with 31 reports, followed by IP protection and Performance. The least observed NFRs are Fault tolerance, Hierarchy and Extensibility. Fault tolerance is especially important for long running co-simulations. One of the industrial partners of the INTO-CPS project has running co-simulations that takes a minimum of two weeks to complete. We argue that Extensibility (the ability to easily accomodate new features) should be given more importance: if an heterogeneous set of SUs participate in the same co-simulation scenario, the combination of capabilities provided (see Fig. 8) can be huge. Thus, the orchestrator can either assume a common homogeneous set of capabilities, which is the most common approach, or can leverage the capabilities provided by each one. In any case, extensibility and hierarchy are crucial to address, and implement, new semantic adaptations.

As Fig. 11 suggests, we could not find approaches that make use of the nominal values of state and output variables, even though these are capabilities supported in the FMI Standard, and are useful to detect invalid co-simulations. A-causal approaches are important for modularity, as explained in Section 4.3, but these are scarce too.

[11] http://msdl.cs.mcgill.ca/people/claudio/pub/Gomes2016bClassificationRawData/raw_data.zip

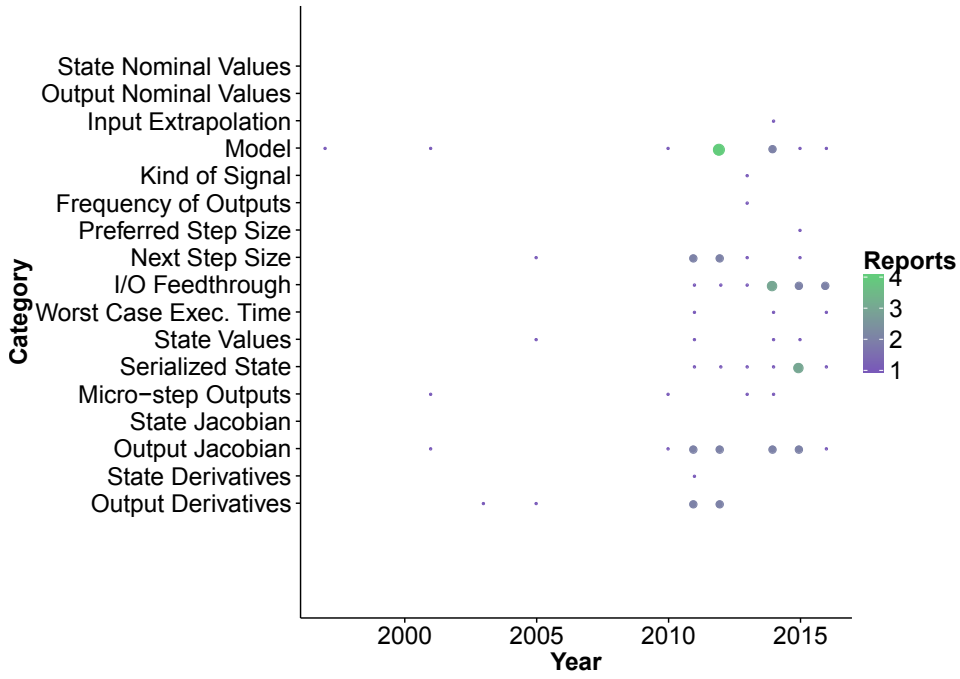


Fig. 11. Classification with respect to SU requirements: information exposed.

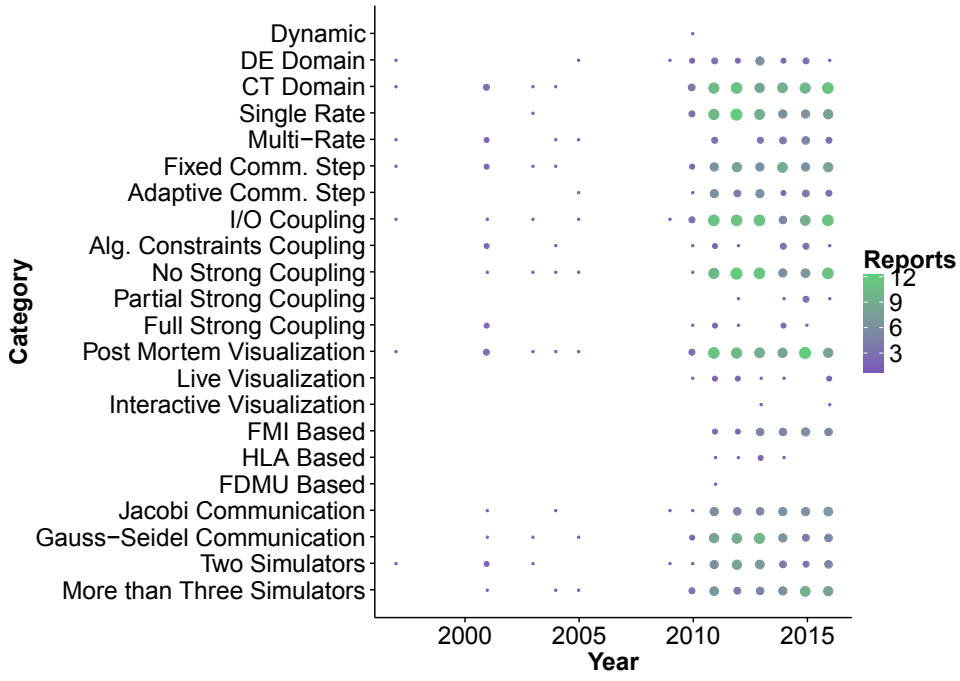


Fig. 12. Classification with respect to framework requirements.

As for the framework requirements, in Fig. 12, the least observed features are dynamic structure co-simulation, interactive visualization, multi-rate, algebraic coupling, and partial/-full strong coupling support. This can be explained by the fact that these features depend upon the capabilities of the SUs, which may not be mature.

Figs. 10 – 12 do not tell the full story because they isolate each feature. Feature interaction is a common phenomenon, and among many possible interactions, we highlight the accuracy concern, domain of the co-simulation, number of SUs supported, and IP protection. As can be seen from Fig. 14, there is only one approach [108] that is both CT and DE based, up to any number of SUs. Accommodating the different CT and DE domains means that the approach assumes that the SUs can behave both as a CT and as a DE SU.

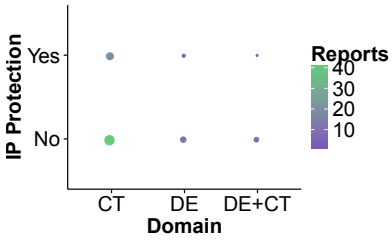


Fig. 13. Formalisms vs IP Protection.

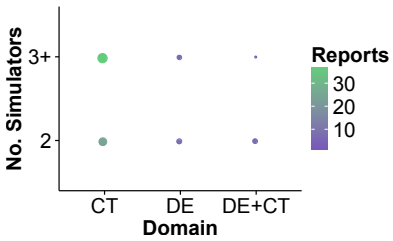


Fig. 14. Formalisms vs SUs.

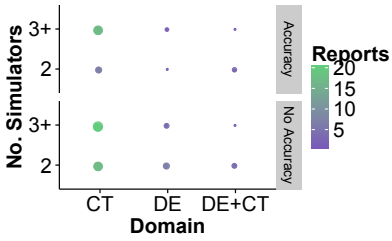


Fig. 15. Accuracy vs Formalisms vs SUs.

The concern with IP protection is evident in Fig. 10 but the number of DE and CT based approaches that provide some support for it is small, as shown in Fig. 13. Similarly, as Fig. 15 suggests, accuracy does not show up a lot in the DE and CT approaches, for more than two SUs. Accuracy is particularly important in interactions between DE and CT SUs.

In general, from the observed classification, there is a lack of research into approaches that are both DE and CT based, and that leverage the extra features from the SUs.

7 CONCLUDING REMARKS

In this overview article, we show that there are many interesting challenges to be explored in co-simulation, which will play a key role in enabling the virtual development of complex heterogeneous systems in the decades to come. The early success can be attributed to a large number of reported applications. However, the large majority of these applications represent *ad-hoc* couplings between two simulators of two different domains (e.g., a network simulator with a power grid one, or a HVAC simulator with a building envelop one)[12]. As systems become increasingly complex, the demand for co-simulation scenarios that are large, hierarchical, heterogeneous, accurate, IP protected, and so on, will increase.

This survey covers the main challenges in enabling co-simulation. To tackle such a broad topic, we have covered two main domains—continuous-time- and discrete-event-based co-simulation—separately and then discussed the challenges that arise when the two domains are combined. A taxonomy is proposed and a classification of the works related to co-simulation in the last five years is carried out using that taxonomy.

From the challenges we highlight: semantic adaptation, modular coupling, stability and accuracy, and finding a standard for hybrid co-simulation. For early system analysis, the adaptations required to combine simulators from different formalisms, even conforming to the same standard, are very difficult to generalize to any co-simulation scenario.

One of the main conclusions of the classification is that there is lack of research into modular, stable, and accurate coupling of simulators in dynamic structure scenarios. This is where acausal approaches for co-simulation can play a key role. The use of bi-directional

[12]We did not consider the (potentially many) unreported applications of co-simulation.

effort/flow ports can be a solution inspired by Bond-graphs [6], and there is some work already in this direction [51].

Finally, this document is an attempt to summarize, bridge, and enhance the future research in co-simulation, wherever it may lead us to.

ACKNOWLEDGMENTS

The authors wish to thank Yentl Van Tendeloo, for the in depth review of DE based co-simulation, Kenneth Lausdahl for providing valuable input and discussions throughout the making of this survey, and TWT GmbH for the valuable input on everyday challenges faced by a co-simulation master. This research was partially supported by Flanders Make vzw, the strategic research centre for the manufacturing industry, and a PhD fellowship grant from the Agency for Innovation by Science and Technology in Flanders (IWT) (dossier 151067). In addition, the work presented here is partially supported by the INTO-CPS project funded by the European Commission's Horizon 2020 programme under grant agreement number 664047 (http://cordis.europa.eu/project/rcn/194142_en.html). This work is also financially supported by the Swedish Foundation for Strategic Research (project FFL15-0032).

BOOKS AND BOOK CHAPTERS

[1] François Edouard Cellier and Ernesto Kofman. 2006. *Continuous System Simulation*. Springer Science & Business Media.

[2] Paul Adrien Maurice Dirac. 1981. *The principles of quantum mechanics*. Number 27. Oxford university press.

[3] Kelly S. Hale and Kay M. Stanney. 2014. *Handbook of virtual environments: Design, implementation, and applications*. CRC Press.

[4] Raphaël Jungers. 2009. *The joint spectral radius: theory and applications*. Vol. 385. Springer Science & Business Media.

[5] Alexander Kossiakoff, William N. Sweet, Samuel J. Seymour, and Steven M. Biemer. 2011. Structure of Complex Systems. In *Systems Engineering Principles and Practice*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 41–67. https://doi.org/10.1002/9781118001028.ch3

[6] Henry M. Paynter. 1961. *Analysis and design of engineering systems*. MIT press.

[7] Claudius Ptolemaeus. 2014. *System Design, Modeling, and Simulation: Using Ptolemy II*. Berkeley: Ptolemy.org.

[8] Arjan J. Van Der Schaft and Johannes Maria Schumacher. 2000. *An introduction to hybrid dynamical systems*. Vol. 251. Springer London.

[9] G. Wanner and E. Hairer. 1991. *Solving ordinary differential equations I: Nonstiff Problems* (springer s ed.). Vol. 1. Springer-Verlag.

[10] Bernard P. Zeigler. 1976. *Theory of modelling and simulation*. New York, Wiley. 435 pages.

[11] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems* (2 ed.). Academic press.

JOURNAL PUBLICATIONS

[12] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. 1995. The algorithmic analysis of hybrid systems. *Theoretical Computer Science* 138, 1 (feb 1995), 3–34. https://doi.org/10.1016/0304-3975(94)00202-T

[13] Andrés A. Alvarez Cabrera, Krijn Woestenenk, and Tetsuo Tomiyama. 2011. An architecture model to support cooperative design for mechatronic products: A control design case. *Mechatronics* 21, 3 (apr 2011), 534–547. https://doi.org/10.1016/j.mechatronics.2011.01.009

[14] Martin Arnold. 2010. Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models. *Journal of Computational and Nonlinear Dynamics* 5, 3 (may 2010), 9. https://doi.org/10.1115/1.4001389

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

[16] Martin Arnold, Stefan Hante, and Markus A Köbis. 2014. Error analysis for co-simulation with force-displacement coupling. *PAMM* 14, 1 (dec 2014), 43–44. https://doi.org/10.1002/pamm.201410014
[17] Fernando J. Barros. 1997. Modeling formalisms for dynamic structure systems. *ACM Transactions on Modeling and Computer Simulation* 7, 4 (oct 1997), 501–515. https://doi.org/10.1145/268403.268423
[18] Paul I. Barton and C. C. Pantelides. 1994. Modeling of combined discrete/continuous processes. *AIChE Journal* 40, 6 (jun 1994), 966–979. https://doi.org/10.1002/aic.690400608
[19] Abir Ben Khaled-El Feki, Laurent Duval, Cyril Faure, Daniel Simon, and Mongi Ben Gaid. 2017. CHOPtrey: contextual online polynomial extrapolation for enhanced multi-core co-simulation of complex systems. *SIMULATION* 93, 3 (jan 2017). https://doi.org/10.1177/0037549716684026
[20] Albert Benveniste, Benoît Caillaud, and Paul Le Guernic. 2000. Compositionality in Dataflow Synchronous Languages: Specification and Distributed Code Generation. *Information and Computation* 163, 1 (nov 2000), 125–171. https://doi.org/10.1006/inco.2000.9999
[21] Massimo Bombino and Patrizia Scandurra. 2013. A model-driven co-simulation environment for heterogeneous systems. *International Journal on Software Tools for Technology Transfer* 15, 4 (aug 2013), 363–374. https://doi.org/10.1007/s10009-012-0230-5
[22] M. S. Branicky, V. S. Borkar, and S. K. Mitter. 1998. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automat. Control* 43, 1 (1998), 31–45. https://doi.org/10.1109/9.654885
[23] Martin Busch. 2016. Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. *ZAMM - Journal of Applied Mathematics and Mechanics* 96, 9 (sep 2016), 1061–1081. https://doi.org/10.1002/zamm.201500196
[24] Martin Busch and Bernhard Schweizer. 2012. Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit coupling approach. *Archive of Applied Mechanics* 82, 6 (jun 2012), 723–741. https://doi.org/10.1007/s00419-011-0586-0
[25] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. 2006. Languages and Tools for Hybrid Systems Design. *Foundations and Trends® in Electronic Design Automation* 1, 1/2 (2006), 1–193. https://doi.org/10.1561/1000000001
[26] François Edouard Cellier. 1977. Combined Continuous/Discrete System Simulation Languages: Usefulness, Experiences and Future Development. *Special Interest Group (SIG) on SImulation and Modeling (SIM)* 9, 1 (1977), 18–21. https://doi.org/10.1145/1102505.1102514
[27] K.M. Chandy and J Misra. 1979. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering* SE-5, 5 (sep 1979), 440–452. https://doi.org/10.1109/TSE.1979.230182
[28] Bo-Chiuan Chen and Huei Peng. 2001. Differential-Braking-Based Rollover Prevention for Sport Utility Vehicles with Human-in-the-loop Evaluations. *Vehicle System Dynamics* 36, 4-5 (aug 2001), 359–389. https://doi.org/10.1076/vesd.36.4.359.3546
[29] Fabio Cremona, Marten Lohstroh, David Broman, Edward A. Lee, Michael Masin, and Stavros Tripakis. 2017. Hybrid co-simulation: it's about time. *Software & Systems Modeling* (nov 2017). https://doi.org/10.1007/s10270-017-0633-6
[30] Joachim Denil, Paul De Meulenaere, Serge Demeyer, and Hans Vangheluwe. 2017. DEVS for AUTOSAR-based system deployment modeling and simulation. *SIMULATION* 93, 6 (jun 2017), 489–513. https://doi.org/10.1177/0037549716684552 arXiv:arXiv:1508.04886v1
[31] Bei Gu and H. Harry Asada. 2004. Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models. *Journal of Dynamic Systems, Measurement, and Control* 126, 1 (apr 2004), 1. https://doi.org/10.1115/1.1648307
[32] Irene Hafner, Bernhard Heinzl, and Matthias Roessler. 2013. An Investigation on Loose Coupling Co-Simulation with the BCVTB. *SNE Simulation Notes Europe* 23, 1 (2013). https://doi.org/10.11128/sne.23.tn.10173
[33] David R. Jefferson. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems* 7, 3 (jul 1985), 404–425. https://doi.org/10.1145/3916.3988
[34] Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. 1999. On the regularization of Zeno hybrid automata. *Systems & Control Letters* 38, 3 (oct 1999), 141–150. https://doi.org/10.1016/S0167-6911(99)00059-6
[35] Rudolph Emil Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* 82, 1 (mar 1960), 35. https://doi.org/10.1115/1.3662552
[36] Tamas Kalmar-Nagy and Ilinca Stanciulescu. 2014. Can complex systems really be simulated? *Appl. Math. Comput.* 227 (jan 2014), 199–211. https://doi.org/10.1016/j.amc.2013.11.037

[37] Ernesto Kofman. 2002. A Second-Order Approximation for DEVS Simulation of Continuous Systems. *SIMULATION* 78, 2 (feb 2002), 76–89. https://doi.org/10.1177/0037549702078002206
[38] Ernesto Kofman and Sergio Junco. 2001. Quantized-state systems: a DEVS Approach for continuous system simulation. *Transactions of The Society for Modeling and Simulation International* 18, 3 (2001), 123–132.
[39] R. Kübler and W. Schiehlen. 2000. Modular Simulation in Multibody System Dynamics. *Multibody System Dynamics* 4, 2-3 (aug 2000), 107–127. https://doi.org/10.1023/A:1009810318420
[40] R. Kübler and W. Schiehlen. 2000. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems* 6, 2 (jun 2000), 93–113. https://doi.org/10.1076/1387-3954(200006)6:2;1-M;FT093
[41] Leslie Lamport. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (jul 1978), 558–565. https://doi.org/10.1145/359545.359563
[42] Bu-Sung Lee, Wentong Cai, Stephen J. Turner, and L. Chen. 2000. Adaptive dead reckoning algorithms for distributed interactive simulation. *International Journal of Simulation* 1, 1-2 (dec 2000), 21–34.
[43] Oded Maler, Zohar Manna, and Amir Pnueli. 1992. From timed to hybrid systems. *Real-Time: Theory in Practice* 600 (1992), 447–484. https://doi.org/10.1007/BFb0032003
[44] Pieter J. Mosterman and Gautam Biswas. 1998. A theory of discontinuities in physical system models. *Journal of the Franklin Institute* 335, 3 (apr 1998), 401–439. https://doi.org/10.1016/S0016-0032(96)00126-3
[45] P. J. Mosterman and Hans Vangheluwe. 2004. Computer Automated Multi-Paradigm Modeling: An Introduction. *Simulation* 80, 9 (sep 2004), 433–450. https://doi.org/10.1177/0037549704050532
[46] Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Jan Peleska. 2015. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *Comput. Surveys* 48, 2 (sep 2015), 18:1—-18:41. https://doi.org/10.1145/2794381
[47] Henrik Nilsson. 2003. Functional automatic differentiation with Dirac impulses. *ACM SIGPLAN Notices* 38, 9 (sep 2003), 153–164. https://doi.org/10.1145/944746.944720
[48] James Nutaro, Phani Teja Kuruganti, Vladimir Protopopescu, and Mallikarjun Shankar. 2012. The split system approach to managing time in simulations of hybrid systems having continuous and discrete event components. *SIMULATION* 88, 3 (mar 2012), 281–298. https://doi.org/10.1177/0037549711401000
[49] Seaseung Oh and Suyong Chae. 2016. A Co-Simulation Framework for Power System Analysis. *Energies* 9, 3 (2016), 131.
[50] Davide Quaglia, Riccardo Muradore, Roberto Bragantini, and Paolo Fiorini. 2012. A SystemC/Matlab co-simulation tool for networked control systems. *Simulation Modelling Practice and Theory* 23 (apr 2012), 71–86. https://doi.org/10.1016/j.simpat.2012.01.003
[51] Severin Sadjina, Lars T. Kyllingstad, Stian Skjong, and Eilif Pedersen. 2017. Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation. *Engineering with Computers* 33, 3 (jul 2017), 607–620. https://doi.org/10.1007/s00366-016-0492-8
[52] Dieter Schramm, Wildan Lalo, and Michael Unterreiner. 2010. Application of Simulators and Simulation Tools for the Functional Design of Mechatronic Systems. *Solid State Phenomena* 166-167 (sep 2010), 1–14. https://doi.org/10.4028/www.scientific.net/SSP.166-167.1
[53] Schweizer and Daixing Lu. 2015. Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 95, 9 (sep 2015), 911–938. https://doi.org/10.1002/zamm.201300191
[54] Bernhard Schweizer, Pu Li, and Daixing Lu. 2015. Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches. *Journal of Computational and Nonlinear Dynamics* 10, 5 (sep 2015), 051007. https://doi.org/10.1115/1.4028503
[55] Bernhard Schweizer, Daixing Lu, and Pu Li. 2016. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. *Multibody System Dynamics* 36, 1 (jan 2016), 1–36. https://doi.org/10.1007/s11044-015-9464-9
[56] S. Sicklinger, V. Belsky, B. Engelmann, H. Elmqvist, H. Olsson, R. Wüchner, and K. U. Bletzinger. 2014. Interface Jacobian-based Co-Simulation. *Internat. J. Numer. Methods Engrg.* 98, 6 (may 2014), 418–444. https://doi.org/10.1002/nme.4637
[57] Georg Stettinger, Martin Benedikt, Martin Horn, Josef Zehetner, and Clenn Giebenhain. 2017. Control of a magnetic levitation system with communication imperfections: A model-based coupling approach. *Control Engineering Practice* 58 (jan 2017), 161–170. https://doi.org/10.1016/j.conengprac.2016.10.009
[58] Brook Taylor. 1715. Methodus Incrementorum Directa et Inversa. *London* (1715).

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

[60] Mamadou K. Traoré and Alexandre Muzy. 2006. Capturing the dual relationship between simulation models and their context. *Simulation Modelling Practice and Theory* 14, 2 (feb 2006), 126–142. https://doi.org/10.1016/j.simpat.2005.03.002

[61] Herman Van der Auweraer, Jan Anthonis, Stijn De Bruyne, and Jan Leuridan. 2013. Virtual engineering at work: the challenges for designing mechatronic products. *Engineering with Computers* 29, 3 (2013), 389–408. https://doi.org/10.1007/s00366-012-0286-6

[62] Hans Vangheluwe. 2008. Foundations of Modelling and Simulation of Complex Systems. *Electronic Communications of the EASST* 10 (2008). https://doi.org/10.14279/tuj.eceasst.10.162.148

[63] A. Verhoeven, B. Tasic, T. G. J. Beelen, E. J. W. ter Maten, and R. M. M. Mattheij. 2008. BDF compound-fast multirate transient analysis with adaptive stepsize control. *Journal of numerical analysis, Industrial and Applied Mathematics* 3, 3-4 (jan 2008), 275–297.

[64] Michael Wetter. 2010. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation* 4, 3 (nov 2010), 185–203. https://doi.org/10.1080/19401493.2010.518631

[65] Ming-chin Wu and Ming-chang Shih. 2003. Simulated and experimental study of hydraulic anti-lock braking system using sliding-mode PWM control. *Mechatronics* 13, 4 (may 2003), 331–351. https://doi.org/10.1016/S0957-4158(01)00049-6

CONFERENCE & WORKSHOP PUBLICATIONS

[66] Martin Arnold, Christoph Clauß, and Tom Schierz. 2014. Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0. In *Progress in Differential-Algebraic Equations*, Sebastian Schöps, Andreas Bartel, Michael Günther, W. E. Jan ter Maten, and C. Peter Müller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 107–125. https://doi.org/10.1007/978-3-662-44926-4_6

[67] Muhammad Usman Awais, Wolfgang Mueller, Atiyah Elsheikh, Peter Palensky, and Edmund Widl. 2013. Using the HLA for Distributed Continuous Simulations. In *8th EUROSIM Congress on Modelling and Simulation*. IEEE, Cardiff, UK, 544–549. https://doi.org/10.1109/EUROSIM.2013.96

[68] Muhammad Usman Awais, Peter Palensky, Atiyah Elsheikh, Edmund Widl, and Stifter Matthias. 2013. The high level architecture RTI as a master to the functional mock-up interface components. In *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, San Diego, USA, 315–320. https://doi.org/10.1109/ICCNC.2013.6504102

[69] Jens Bastian, Christoph Clauß, Susann Wolf, and Peter Schneider. 2011. Master for Co-Simulation Using FMI. In *8th International Modelica Conference*. Dresden, Germany, 115–120. https://doi.org/10.3384/ecp11063115

[70] Abir Ben Khaled, Laurent Duval, Mohamed El Mongi Ben Gaïd, and Daniel Simon. 2014. Context-based polynomial extrapolation and slackened synchronization for fast multi-core simulation using FMI. In *10th International Modelica Conference*. Linköping University Electronic Press, 225–234.

[71] Torsten Blochwitz, Martin Otter, Martin Arnold, C. Bausch, Christoph Clauss, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, M. Monteiro, T. Neidhold, Dietmar Neumerkel, Hans Olsson, J.-V. Peetz, and S. Wolf. 2011. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *8th International Modelica Conference*. Linköping University Electronic Press; Linköpings universitet, Dresden, Germany, 105–114. https://doi.org/10.3384/ecp11063105

[72] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. 2012. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International Modelica Conference*. Linköping University Electronic Press, Munich, Germany, 173–184. https://doi.org/10.3384/ecp12076173

[73] Sergiy Bogomolov, Marius Greitschus, Peter G. Jensen, Kim G. Larsen, Marius Mikučionis, Thomas Strump, and Stavros Tripakis. 2015. Co-Simulation of Hybrid Systems with SpaceEx and Uppaal. In *11th International Modelica Conference*. Linköping University Electronic Press, Paris, France, 159–169. https://doi.org/10.3384/ecp15118159

[74] Jean-Sébastien Bolduc and Hans Vangheluwe. 2002. Expressing ODE models as DEVS: Quantization approaches.. In *AI, Simulation and Planning in High Autonomy Systems*, F. Barros and N. Giambiasi

(Eds.). Lisbon, Portugal, 163–169.
[75] Jean-Sébastien Bolduc and Hans Vangheluwe. 2003. Mapping ODES to DEVS: Adaptive quantization. In *Summer Computer Simulation Conference*. Society for Computer Simulation International, Montreal, Quebec, Canada, 401–407.
[76] F. Bouchhima, M. Brière, G Nicolescu, M Abid, and E. Aboulhamid. 2006. A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation. In *IEEE International Behavioral Modeling and Simulation Workshop*. IEEE, 1–6. https://doi.org/10.1109/BMAS.2006.283461
[77] Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Dominique Marcadet. 2011. Semantic Adaptation for Models of Computation. In *11th International Conference on Application of Concurrency to System Design (ACSD)*. 153–162. https://doi.org/10.1109/ACSD.2011.17
[78] Jonathan Brembeck, Andreas Pfeiffer, Michael Fleps-Dezasse, Martin Otter, Karl Wernersson, and Hilding Elmqvist. 2014. Nonlinear State Estimation with an Extended FMI 2.0 Co-Simulation Interface. In *10th International Modelica Conference*. Linköping University Electronic Press; Linköpings universitet, 53–62. https://doi.org/10.3384/ecp1409653
[79] David Broman, Christopher Brooks, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. 2013. Determinate composition of FMUs for co-simulation. In *Eleventh ACM International Conference on Embedded Software*. IEEE Press Piscataway, NJ, USA, Montreal, Quebec, Canada.
[80] David Broman, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. 2015. Requirements for Hybrid Cosimulation Standards. In *18th International Conference on Hybrid Systems: Computation and Control (HSCC '15)*. ACM, New York, NY, USA, 179–188. https://doi.org/10.1145/2728606.2728629
[81] David Broman, Edward A. Lee, Stavros Tripakis, and Martin Törngren. 2012. Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*. ACM, 49–54. https://doi.org/10.1145/2508443.2508452
[82] Martin Busch and Bernhard Schweizer. 2010. Numerical stability and accuracy of different co-simulation techniques: analytical investigations based on a 2-DOF test model. In *1st Joint International Conference on Multibody System Dynamics*. 25–27.
[83] Martin Busch and Bernhard Schweizer. 2011. An explicit approach for controlling the macro-step size of co-simulation methods. In *7th European Nonlinear Dynamics*. Rome, Italy, 24–29.
[84] Martin Busch and Bernhard Schweizer. 2011. Stability of Co-Simulation Methods Using Hermite and Lagrange Approximation Techniques. In *ECCOMAS Thematic Conference on Multibody Dynamics*. Brussels, Belgium, 1–10.
[85] Benjamin Camus, Christine Bourjot, and Vincent Chevrier. 2015. Combining DEVS with multi-agent concepts to design and simulate multi-models of complex systems (WIP). In *Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*. Society for Computer Simulation International, 85–90.
[86] Benjamin Camus, Virginie Galtier, Mathieu Caujolle, Vincent Chevrier, Julien Vaubourg, Laurent Ciarletta, and Christine Bourjot. 2016. Hybrid Co-simulation of FMUs using DEV&DESS in MECSYCO. In *Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium (TMS/DEVS 16)*. Pasadena, CA, United States.
[87] Alex Chung Hen Chow and Bernard P. Zeigler. 1994. Parallel DEVS: A Parallel, Hierarchical, Modular, Modeling Formalism. In *26th Conference on Winter Simulation (WSC '94)*. Society for Computer Simulation International, San Diego, CA, USA, 716–722.
[88] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A Lee. 2016. FIDE: an FMI integrated development environment. In *31st Annual ACM Symposium on Applied Computing (SAC '16)*. ACM Press, New York, New York, USA, 1759–1766. https://doi.org/10.1145/2851613.2851677
[89] Joachim Denil, Bart Meyers, Paul De Meulenaere, and Hans Vangheluwe. 2015. Explicit Semantic Adaptation of Hybrid Formalisms for FMI Co-Simulation. In *Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, Society for Computer Simulation International (Ed.). Alexandria, Virginia, 99–106.
[90] Sven Dronka and Jochen Rauh. 2006. Co-simulation-interface for user-force-elements. In *SIMPACK user meeting*. Baden-Baden.
[91] Olaf Enge-Rosenblatt, Christoph Clauß, André Schneider, and Peter Schneider. 2011. Functional Digital Mock-up and the Functional Mock-up Interface–Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems. In *8th International Modelica Conference*. Linköping University

Electronic Press; Linköpings universitet, Dresden, Germany.
[92] Yishai A. Feldman, Lev Greenberg, and Eldad Palachi. 2014. Simulating Rhapsody SysML Blocks in Hybrid Models with FMI. In *10th International Modelica Conference*. Linköping University Electronic Press, 43–52. https://doi.org/10.3384/ecp1409643
[93] P. Fey, H.W. Carter, and P.A. Wilsey. 1997. Parallel synchronization of continuous time discrete event simulators. In *International Conference on Parallel Processing (Cat. No.97TB100162)*. IEEE Comput. Soc, 227–231. https://doi.org/10.1109/ICPP.1997.622649
[94] Jonathan Friedman and Jason Ghidella. 2006. Using Model-Based Design for Automotive Systems Engineering - Requirements Analysis of the Power Window Example. SAE Technical Paper. https://doi.org/10.4271/2006-01-1217
[95] Richard M. Fujimoto. 2001. Parallel and distributed simulation systems. In *Winter Simulation Conference (Cat. No.01CH37304)* (1 ed.), Vol. 300. Wiley New York, Arlington, VA, USA, 147–157. https://doi.org/10.1109/WSC.2001.977259
[96] Virginie Galtier, Stephane Vialle, Cherifa Dad, Jean-Philippe Tavella, Jean-Philippe Lam-Yee-Mui, and Gilles Plessis. 2015. FMI-Based Distributed Multi-Simulation with DACCOSIM. In *Spring Simulation Multi-Conference*. Society for Computer Simulation International, Alexandria, Virginia, USA, 804–811.
[97] Alfredo Garro and Alberto Falcone. 2015. On the integration of HLA and FMI for supporting interoperability and reusability in distributed simulation. In *Spring Simulation Multi-Conference*. Society for Computer Simulation International, 774–781.
[98] A. Ghosh, M. Bershteyn, R. Casley, C. Chien, A. Jain, M. Lipsie, D. Tarrodaychik, and O. Yamamoto. 1995. A hardware-software co-simulator for embedded system design and debugging. In *Design Automation Conference*. Chiba, Japan, 155–164. https://doi.org/10.1109/ASPDAC.1995.486217
[99] Edward Glaessgen and David Stargel. 2012. 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*. American Institute of Aeronautics and Astronautics, Reston, Virigina, 1–14. https://doi.org/10.2514/6.2012-1818
[100] Cláudio Gomes. 2016. Foundations for Continuous Time Hierarchical Co-simulation. In *ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems)*. Saint Malo, Brittany, France, to appear.
[101] Cláudio Gomes, Paschalis Karalis, Eva M. Navarro-López, and Hans Vangheluwe. 2018. Approximated Stability Analysis of Bi-modal Hybrid Co-simulation Scenarios. In *1st Workshop on Formal Co-Simulation of Cyber-Physical Systems*. Springer, Cham, Trento, Italy, 345–360. https://doi.org/10.1007/978-3-319-74781-1_24
[102] Cláudio Gomes, Benoît Legat, Raphaël M. Jungers, and Hans Vangheluwe. 2017. Stable Adaptive Co-simulation : A Switched Systems Approach. In *IUTAM Symposium on Co-Simulation and Solver Coupling*. Darmstadt, Germany, to appear.
[103] Cláudio Gomes, Yentl Van Tendeloo, Joachim Denil, Paul De Meulenaere, and Hans Vangheluwe. 2017. Hybrid System Modelling and Simulation with Dirac Deltas. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (DEVS '17)*. Society for Computer Simulation International, Virginia Beach, Virginia, USA, Article No. 7.
[104] Bei Gu and H. Harry Asada. 2001. Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference*, Vol. 3. Arlington, VA, USA, 2273–2278. https://doi.org/10.1109/ACC.2001.946089
[105] Andreas Himmler. 2013. Hardware-in-the-Loop Technology Enabling Flexible Testing Processes. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Grapevine (Dallas/Ft. Worth Region), Texas, 1–8. https://doi.org/10.2514/6.2013-816
[106] Velin Kounev, David Tipper, Martin Levesque, Brandon M. Grainger, Thomas Mcdermott, and Gregory F. Reed. 2015. A microgrid co-simulation framework. In *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, Lévesque, McDermott, 1–6. https://doi.org/10.1109/MSCPES.2015.7115398
[107] Martin Krammer, Johannes Fritz, and Michael Karner. 2015. Model-Based Configuration of Automotive Co-Simulation Scenarios. In *48th Annual Simulation Symposium*. Society for Computer Simulation International, Alexandria, Virginia, 155–162.
[108] T. Kuhr, T. Forster, T. Braun, and R. Gotzhein. 2013. FERAL - Framework for simulator coupling on requirements and architecture level. In *Eleventh IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*. IEEE, Portland, OR, USA, 11–22.

[109] David P. Y. Lawrence, Cláudio Gomes, Joachim Denil, Hans Vangheluwe, and Didier Buchs. 2016. Coupling Petri nets with Deterministic Formalisms Using Co-simulation. In *Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*. Pasadena, CA, USA, 6:1—-6:8.

[110] Edward A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. 363–369. https://doi.org/10.1109/ISORC.2008.25

[111] Edward A. Lee and Haiyang Zheng. 2005. Operational semantics of hybrid systems. In *Hybrid Systems: Computation and Control (Lecture Notes in Computer Science)*, Manfred Morari and Lothar Thiele (Eds.), Vol. 3414. Springer Berlin Heidelberg, 25–53. https://doi.org/10.1007/978-3-540-31954-2_2

[112] E. Lelarasmee, Albert E. Ruehli, and A. L. Sangiovanni-Vincentelli. 1982. The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 1. 131–145. https://doi.org/10.1109/TCAD.1982.1270004

[113] Zohar Manna and Amir Pnueli. 1993. Verifying hybrid systems. In *Hybrid Systems SE - 2 (Lecture Notes in Computer Science)*, Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel (Eds.), Vol. 736. Springer Berlin Heidelberg, 4–35. https://doi.org/10.1007/3-540-57318-6_22

[114] Bart Meyers, Joachim Denil, Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet, and Hans Vangheluwe. 2013. A DSL for Explicit Semantic Adaptation. In *7th International Workshop on Multi-Paradigm Modeling (CEUR Workshop Proceedings)*, Edward Jones Tamás Mészáros Christophe Jacquet Daniel Balasubramanian (Ed.). Miami, United States, 47–56.

[115] Pieter J. Mosterman. 1999. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In *Hybrid Systems: Computation and Control SE - 17 (Lecture Notes in Computer Science)*, Frits W. Vaandrager and Jan H. van Schuppen (Eds.), Vol. 1569. Springer Berlin Heidelberg, Berg en Dal, The Netherlands, 165–177. https://doi.org/10.1007/3-540-48983-5_17

[116] W. Müller and E. Widl. 2015. Using FMI components in discrete event systems. In *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. Seattle, WA, USA, 1–6. https://doi.org/10.1109/MSCPES.2015.7115397

[117] Sadaf Mustafiz, Bruno Barroca, Cláudio Gomes, and Hans Vangheluwe. 2016. Towards Modular Language Design Using Language Fragments: The Hybrid Systems Case Study. In *13th International Conference on Information Technology - New Generations (ITNG)*, Shahram Latifi (Ed.). Springer, Cham, Las Vegas, NV USA, 785–797. https://doi.org/10.1007/978-3-319-32467-8_68

[118] Sadaf Mustafiz and Hans Vangheluwe. 2013. Explicit Modelling of Statechart Simulation Environments. In *Summer Computer Simulation Conference (SCSC '13)*. Society for Modeling & Simulation International, Vista, CA, 21:1—-21:8.

[119] Alexandre Muzy, Luc Touraille, Hans Vangheluwe, Olivier Michel, Mamadou Kaba Traoré, and David R. C. Hill. 2010. Activity Regions for the Specification of Discrete Event Systems. In *Spring Simulation Multiconference*. Society for Computer Simulation International, San Diego, CA, USA, 136:1—-136:7. https://doi.org/10.1145/1878537.1878679

[120] Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh, Hubertus Tummescheit, and Chandrasekar Sureshkumar. 2014. Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems. In *10th International Modelica Conference*. 10–12.

[121] Kristoffer Norling, David Broman, Peter Fritzson, Alexander Siemers, and Dag Fritzson. 2007. Secure distributed co-simulation over wide area networks. In *48th Conference on Simulation and Modelling*. Göteborg, Sweden, 14–23.

[122] James Nutaro. 2011. Designing power system simulators for the smart grid: Combining controls, communications, and electro-mechanical dynamics. In *IEEE Power and Energy Society General Meeting*. IEEE, Detroit, MI, USA, 1–5. https://doi.org/10.1109/PES.2011.6039456

[123] James Nutaro. 2016. A method for bounding error in multi-rate and federated simulations. In *Winter Simulation Conference*. IEEE, Washington, DC, USA, 967–976. https://doi.org/10.1109/WSC.2016.7822157

[124] Nicolai Pedersen, Tom Bojsen, and Jan Madsen. 2017. Co-simulation of Cyber Physical Systems with HMI for Human in the Loop Investigations. In *Symposium on Theory of Modeling & Simulation (TMS/DEVS '17)*. Society for Computer Simulation International, Virginia Beach, Virginia, USA, 1:1—-1:12. http://dl.acm.org/citation.cfm?id=3108905.3108906

[125] Nicolai Pedersen, Tom Bojsen, Jan Madsen, and Morten Vejlgaard-Laursen. 2016. FMI for Co-Simulation of Embedded Control Software. In *The First Japanese Modelica Conferences*. Linköping

University Electronic Press, Tokyo, Japan, 70–77. https://doi.org/10.3384/ecp1612470

[126] Nicolai Pedersen, Kenneth Lausdahl, Enrique Vidal Sanchez, Peter Gorm Larsen, and Jan Madsen. 2017. 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*. SCITEPRESS - Science and Technology Publications, 73–82. https://doi.org/10.5220/0006412700730082

[127] Régis Plateaux, J.Y. Choley, Olivia Penas, and Alain Riviere. 2009. Towards an integrated mechatronic design process. In *International Conference on Mechatronics*, Vol. 00. IEEE, Malaga, Spain, 1–6. https://doi.org/10.1109/ICMECH.2009.4957237

[128] Gauthier Quesnel, Raphaël Duboz, David Versmisse, and E. Ramat. 2005. DEVS coupling of spatial and ordinary differential equations: VLE framework. In *Open International Conference on Modeling and Simulation*, Vol. 5. Citeseer, 281–294.

[129] G. Stettinger, M. Horn, M. Benedikt, and J. Zehetner. 2014. Model-based coupling approach for non-iterative real-time co-simulation. In *European Control Conference (ECC)*. 2084–2089. https://doi.org/10.1109/ECC.2014.6862242

[130] Georg Stettinger, Josef Zehetner, Martin Benedikt, and Norbert Thek. 2013. Extending Co-Simulation to the Real-Time Domain. In *SAE Technical Paper*. https://doi.org/10.4271/2013-01-0421

[131] Robert Tarjan. 1971. Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, Vol. 1. East Lansing, MI, USA. https://doi.org/10.1109/SWAT.1971.10

[132] Jean-Philippe Tavella, Mathieu Caujolle, Stephane Vialle, Cherifa Dad, Charles Tan, Gilles Plessis, Mathieu Schumann, Arnaud Cuccuru, and Sebastien Revol. 2016. Toward an accurate and fast hybrid multi-simulation with the FMI-CS standard. (sep 2016), 5 pages. https://doi.org/10.1109/ETFA.2016.7733616

[133] Marija Trcka, Michael Wetter, and Jan Hensen. 2007. Comparison of co-simulation approaches for building and HVAC/R system simulation. In *International IBPSA Conference*. Beijing, China.

[134] Stavros Tripakis. 2015. Bridging the semantic gap between heterogeneous modeling formalisms and FMI. In *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. IEEE, 60–69. https://doi.org/10.1109/SAMOS.2015.7363660

[135] Adelinde M. Uhrmacher. 1993. Variable structure models: autonomy and control answers from two different modeling approaches. In *AI, Simulation and Planning in High Autonomy Systems*. IEEE Comput. Soc. Press, 133–139. https://doi.org/10.1109/AIHAS.1993.410588

[136] Bert Van Acker, Joachim Denil, Paul De Meulenaere, and Hans Vangheluwe. 2015. Generation of an Optimised Master Algorithm for FMI Co-simulation. In *Symposium on Theory of Modeling & Simulation-DEVS Integrative*. Society for Computer Simulation International, 946–953.

[137] Simon Van Mierlo. 2015. Explicitly Modelling Model Debugging Environments. In *ACM Student Research Competition (ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems)*. CEUR, 24–29.

[138] Yentl Van Tendeloo and Hans Vangheluwe. 2014. Activity in PythonPDEVS. In *ITM Web of Conferences*, R. Castro, A. Muzy, and L. Capocchi (Eds.), Vol. 3. 10. https://doi.org/10.1051/itmconf/20140301002

[139] Yentl Van Tendeloo and Hans Vangheluwe. 2015. PythonPDEVS: a distributed Parallel DEVS simulator. In *Spring Simulation Multiconference (SpringSim '15)*. Society for Computer Simulation International, Alexandria, Virginia, 844–851.

[140] Hans Vangheluwe. 2000. DEVS as a common denominator for multi-formalism hybrid systems modelling. In *International Symposium on Computer-Aided Control System Design (Cat. No.00TH8537)*. IEEE, Anchorage, AK, USA, 129–134. https://doi.org/10.1109/CACSD.2000.900199

[141] Hans Vangheluwe, Juan De Lara, and Pieter J. Mosterman. 2002. An introduction to multi-paradigm modelling and simulation. In *AI, Simulation and Planning in High Autonomy Systems*. SCS, 9–20.

[142] E. Widl, W. Müller, A. Elsheikh, M. Hörtenhuber, and P. Palensky. 2013. The FMI++ library: A high-level utility package for FMI for model exchange. In *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, Berkeley, CA, USA, 1–6. https://doi.org/10.1109/MSCPES.2013.6623316

[143] Faruk Yılmaz, Umut Durak, Koray Taylan, and Halit Oğuztüzün. 2014. Adapting Functional Mockup Units for HLA-compliant Distributed Simulation. In *10th International Modelica Conference*.

[144] Bernard P. Zeigler. 2006. Embedding DEV&DESS in DEVS. In *DEVS Integrative Modeling & Simulation Symposium*, Vol. 7.

[145] Bernard P. Zeigler and J. S. Lee. 1998. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment, Alex F. Sisti (Ed.), Vol. 3369. 49–58. https://doi.org/10.1117/12.319354
[146] Fu Zhang, Murali Yeddanapudi, and Pieter Mosterman. 2008. Zero-crossing location and detection algorithms for hybrid system simulation. In *IFAC World Congress*. 7967–7972.

TECHNICAL REPORTS & PHD THESIS

[147] Christian Andersson. 2016. *Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface*. Ph.D. Dissertation. Lund University.
[148] Christian Andersson, Claus Führer, and Johan Åkesson. 2016. *Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers*. Technical Report 1. 13 pages. http://lup.lub.lu.se/record/dbaf9c49-b118-4ff9-af2e-e1e3102e5c22
[149] David Broman. 2017. *Hybrid Simulation Safety: Limbos and Zero Crossings*. Technical Report. arXiv:1710.06516 https://arxiv.org/abs/1710.06516
[150] FMI. 2014. *Functional Mock-up Interface for Model Exchange and Co-Simulation*. Technical Report.
[151] Markus Friedrich. 2011. *Parallel Co-Simulation for Mechatronic Systems*. Ph.D. Dissertation.
[152] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. 2017. *Co-simulation: State of the art*. Technical Report. arXiv:1702.00686 http://arxiv.org/abs/1702.00686
[153] Matthias Hoepfer. 2011. *Towards a Comprehensive Framework for Co- Simulation of Dynamic Models With an Emphasis on Time Stepping*. Ph.D. Dissertation.
[154] K. C. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. 1990. *Feature-Oriented Domain Analysis. Feasibility study,*. Technical Report. Carnegie Mellon University. 147 pages.
[155] John Lygeros. 2004. *Lecture notes on hybrid systems*. Technical Report. Department of Electrical and Computer Engineering University of Patras. https://robotics.eecs.berkeley.edu/{~}sastry/ee291e/lygeros.pdf
[156] Yentl Van Tendeloo and Hans Vangheluwe. 2017. *An Introduction to Classic DEVS*. Technical Report. 1–24 pages. arXiv:1701.07697 https://arxiv.org/pdf/1701.07697v1.pdf

OTHER REFERENCES

[157] 2007. Modelica - A Unified Object-Oriented Language for Physical Systems Modeling. (2007), Version 3.0 pages. https://www.modelica.org/documents/ModelicaSpec30.pdf
[158] 2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification. (2010). https://standards.ieee.org/findstds/standard/1516-2010.html

Received February 2017; revised September 2017; accepted January 2018