



TECHNISCHE
UNIVERSITÄT
WIEN

Hybrid Modeling of Production Systems: Co-simulation and DEVS-based Approach

DIPLOMA THESIS

conducted in partial fulfillment of the requirements for the degree of
Diplom-Ingenieur (Dipl.-Ing.)

supervised by
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner

submitted to the
Faculty of Electrical Engineering and Information Technology
Vienna University of Technology

by
Bernhard Heinzl
Braunspurgasse 12/2/6/40
1100 Wien

Wien, November 2016

(Signature of Author)

(Signature of Advisor)

Abstract

When investigating physical systems, there is a growing need to perform increasingly complex and cross-domain computer simulations, which require suited methods to describe hybrid simulation models with both continuous and discrete-time dynamics. For example, a simulation model of a production facility should be able to incorporate production entities (discrete) as well as energy flows (continuous). However, implementing heterogeneous hybrid simulation models is still a challenge. Two alternative approaches for this problem, both of which promise different advantages and drawbacks, are investigated and evaluated in this thesis. The first and more common approach pursues coupling of several simulation tools as part of a co-simulation. This is compared to a novel approach that uses a formal model description based on DEVS (Discrete Event System Specification).

Two comprehensive case studies in the context of interdisciplinary simulation of production facilities for energy efficiency investigations are implemented to demonstrate both modeling approaches in practical application. Both case study models include discrete as well as continuous aspects, reflecting components for production equipment, energy system, building services and the building hull. For implementation, modern state-of-the-art simulation tools from research literature are employed: BCVTB (Building Controls Virtual Test Bed) provides a middleware solution for co-simulation, while the MatlabDEVS Toolbox implements a hybrid DEVS simulator.

Based on the case studies, relevant modeling aspects are examined and compared how these can be implemented using co-simulation and the DEVS-based approach, such as modeling of discrete persistent entities, communication between components and handling of differential equations. A subsequent evaluation provides a direct comparison of both approaches with regard to relevant criteria derived from state-of-the-art literature, including reusability of model components, modularity, support of simulation algorithms and overall modeling effort.

Compared to co-simulation, the DEVS-based approach is able to provide integration of continuous and discrete modeling aspects not just on the data level, but also on the modeling level, which entails several major benefits for model development, including improved modularity of hybrid components, model maintainability and ultimately better reusability. However, DEVS-based modeling currently lacks support for high-level specialized modeling features, thus requiring more effort from model developers for initial implementation.

Kurzfassung

Die Untersuchung technisch-physikalischer Systeme erfordert es heutzutage immer komplexere und interdisziplinäre Computersimulationen durchzuführen, wodurch wiederum geeignete Verfahren zur Beschreibung von hybriden Simulationsmodellen mit kontinuierlicher und diskreter Zeitdynamik erforderlich werden. Beispielsweise soll ein Simulationsmodell eines industriellen Produktionsbetriebes in der Lage sein, sowohl Produktionseinheiten (diskret) als auch Energieströme (kontinuierlich) zu modellieren. Die Implementierung derartiger heterogener Simulationsmodelle ist allerdings noch immer eine große Herausforderung. Zwei alternative Ansätze für dieses Problem, die beide unterschiedliche Vor- und Nachteile versprechen, werden in dieser Arbeit untersucht. Der erste Ansatz verfolgt die Kopplung mehrerer Simulationswerkzeuge als Teil einer Co-Simulation. Diese Möglichkeit wird verglichen mit einem zweiten Ansatz basierend auf einer formalen Beschreibung von hybriden Modellen mittels DEVS (Discrete Event System Specification).

Zwei umfassende Fallstudien aus dem Bereich der interdisziplinären Simulation von Produktionsbetrieben demonstrieren diese Modellierungsansätze in der Praxis. Beide Fallstudien umfassen sowohl diskrete als auch kontinuierliche Aspekte in Form von Produktionsanlagen, Energiesysteme, Gebäudetechnik und Gebäudehülle. Zur Implementierung werden State-of-the-Art-Simulationswerkzeuge eingesetzt: BCVTB (Building Controls Virtual Test Bed) stellt eine Middleware-Lösung für die Co-Simulation zur Verfügung, während die MatlabDEVS-Toolbox einen hybriden DEVS-Simulator implementiert.

Basierend auf den Fallstudien werden relevante Modellierungsaspekte verglichen, speziell die Modellierung diskreter persistenter Entitäten, Kommunikation zwischen Komponenten und Handhabung von Differentialgleichungen. Eine anschließende Evaluierung liefert einen direkten Vergleich beider Ansätze in Bezug auf relevante Kriterien, die aus der Forschungsliteratur bezogen werden, u.a. Wiederverwendbarkeit, Modularität, Unterstützung von einschlägigen Simulationsalgorithmen und Modellierungsaufwand.

Im Vergleich zur Co-Simulation ermöglicht der DEVS-basierte Ansatz eine Integration von kontinuierlichen und diskreten Aspekten nicht nur auf der Datenebene sondern auch auf der Modellebene, was wesentliche Vorteile für die Modellentwicklung mit sich bringt, einschließlich verbesserter Modularität von hybriden Komponenten sowie Wiederverwendbarkeit. Allerdings fehlt es einer DEVS-basierten Modellierung an Unterstützung von spezialisierten Modellcharakteristiken, womit ein deutlich höherer Aufwand seitens der Modellentwickler für die Erstimplementierung erforderlich wird.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Statement	2
1.3	Scope of the Work	3
1.4	Method	3
2	Co-simulation Case Study	5
2.1	Background	5
2.1.1	Terminology	6
2.1.2	Coupling Strategies	9
2.1.3	Technologies and Tools for Co-simulation	13
2.2	Design	14
2.2.1	Reference Model	14
2.2.2	Co-simulation Architecture	16
2.2.3	Data Exchange and Synchronization	20
2.3	Implementation	20
2.3.1	Sub-Models	20
2.3.2	BCVTB Middleware	24
2.3.3	Validation	26
2.4	Testing and Results	26
3	DEVS-based Modeling Case Study	31
3.1	Background	31
3.1.1	DEVS-based System Specification Formalisms	31
3.1.2	Tools for DEVS-based Hybrid Modeling and Simulation	38
3.2	Design	40
3.2.1	Modular Hybrid Modeling Approach	40
3.2.2	Simulation Approach	41
3.3	Implementation	43
3.3.1	Hybrid Model Component: Oven	43
3.3.2	Overall Model	48
3.3.3	Validation	50
3.4	Testing and Results	51

4	Evaluation and Comparison	55
4.1	Related Work	55
4.2	Scope of the Evaluation	57
4.3	Criteria	58
4.3.1	User Considerations	58
4.3.2	Modeling Capabilities	60
4.3.3	Simulation Performance	61
4.4	Evaluation of Co-Simulation	61
4.5	Evaluation of hybrid DEVS-based Modeling	66
4.6	Comparison	69
4.7	Conclusion	71
5	Summary & Outlook	75
5.1	Summary	75
5.2	Future Work	76
A	MatlabDEVS Source Code for the Oven Model	77
	List of Figures	87
	List of Tables	89
	Bibliography	91

Introduction

1.1 Motivation

Energy and resource efficiency is becoming an increasingly important topic in the industrial sector, due to its high energy consumption and associated costs, while at the same time showing significant potential for savings [14], [96]. There is a variety of methods and tools available that all aim at improving individual aspects of energy efficiency within production facilities (e.g. reduction and reuse of waste heat, energy-efficient machinery, reduction of building heating demand). Apart from these selected in-depth investigations, there is a potential for additional energy savings that lies in optimizing interactions and interdependencies of multiple domains, including production, energy infrastructure and building technology. These additional potentials can be made accessible by comprehensive cross-domain analysis of production facilities as a whole. The complexity of this task, however, demands sophisticated technologies and tools to support decision-making and assessment of qualified predictions about the impact of different energy saving measures. This includes not only operation and production scheduling, but planning of new production plants as well, where possibilities for change and their potential financial impact are especially high.

One of the most promising approaches in this regard are simulation-based methodologies [128] that aim at an integrated simulation of interdisciplinary aspects of production facilities [53]. Yet, covering such systems as a whole in terms of a simulation model that includes various domains, ranging from production machinery, energy system, to building services and others, has proven difficult as it has to incorporate continuous as well as discrete modeling aspects, such as energy flows, production entities or control signals [105], resulting in so-called *hybrid models* [37], [132].

Implementing such heterogeneous hybrid simulation models still presents a challenge. One possible approach pursues coupling of multiple simulation tools as part of a so-called *co-simulation* (cooperative simulation). This allows utilizing established mature

simulation environments including model descriptions and simulation algorithms tailored to the respective system domain. Apart from the computational overhead, this type of simulation introduces significant complexity into the modeling process, in particular regarding model development, maintenance and reusability of existing components.

Co-simulation is one of the more common methods for hybrid simulation. Similar methodologies can also be found in other areas, including cyber-physical systems [65], smart grids [47] or embedded systems [5].

Other approaches to hybrid simulation aim at a formal model description, for example based on *DEVS* (Discrete Event System Specification) [142] as a formalization of discrete-event models. While such integrated model descriptions are promising in theory, they remain subject of research [105] and still suffer from lack of available mature hybrid simulation tools, preventing widespread acceptance in the industry.

1.2 Problem Statement

Both the *DEVS* formalism as well as co-simulation require a specific modeling approach that implies certain advantages and drawbacks. While co-simulation is more common in various areas, *DEVS*-based modeling promises potential advancements in terms of model handling, modularity and reusability.

This is especially relevant for hybrid models with both continuous and discrete dynamics, as these types of models pose additional restrictions on a possible implementation. For co-simulation for example, respective discrete or continuous model aspects need to be implemented in a suitable simulation software that is equipped with appropriate computation algorithms.

As it is not clear how the novel *DEVS*-based approach compares to co-simulation in practical application, both methods need to be investigated. In particular, we want to take into consideration the application of interdisciplinary simulation of production facilities in the context of energy efficiency investigations.

This leads to the following research questions, which are to be examined in the scope of this thesis:

Research Question 1: *Is a formal modeling approach based on *DEVS* suited for interdisciplinary modeling and simulation of production systems?*

Research Question 2: *How does the *DEVS*-based approach compare to a common co-simulation solution with regard to interdisciplinary production systems?*

The answers to these questions will allow practitioners to determine which approach is suited better for a particular application scenario by comparing and weighing the possibilities based on relevant criteria.

1.3 Scope of the Work

The aim of this work is to investigate the viability of a DEVS-based approach to hybrid simulation modeling compared to common co-simulation in practice. As a concrete application field, we aim at interdisciplinary simulation of industrial production facilities.

For examining co-simulation and DEVS -based modeling, two case studies are carried out, that demonstrate the differences, advantages and drawbacks of both approaches. A subsequent comparison and qualitative evaluation will analyse advantages and disadvantages of the novel DEVS approach in more detail, and evaluate which model description is more suited for hybrid simulation in industrial environments. Special focus is put less on simulation performance or numerical results than on effort for model development, maintainability and reusability of model components.

This case-study-based investigation combines theoretical research with practical insights from a concrete application, thereby allowing to derive results that could be utilized in practice and provide incentives for further research on the basis of case study experience [57]. For example, the study can potentially provide insights into how appropriate DEVS-based hybrid simulation tools can be improved in the future and what features they would have to offer in order to enable more wide-spread use in practical applications.

1.4 Method

First, relevant background for co-simulation and DEVS-based modeling is studied to gain insights into respective modeling restrictions, simulation capabilities and features. Relevant simulation tools are also examined to provide an overview of the current state of the art.

Afterwards, an independent and comprehensive case study is carried out for each of the considered modeling approaches to demonstrate practical application as well as gain experience and information for the subsequent evaluation.

A survey of relevant literature on evaluation and selection of simulation software will provide criteria, based on which both model descriptions will be evaluated.

Finally, a direct comparison of evaluation results will emphasize advantages and drawbacks of each approach and will allow drawing conclusions in order to answer the research questions specified at the beginning.

Co-simulation Case Study

This chapter presents a case study of a co-simulation for interdisciplinary investigations regarding energy efficiency of production facilities. Specifically, the case study focuses on a production plant for a high-end metal-cutting company performing small series production. The goal was to support the planning process of industrial facilities by providing a tool that allows to compare different design variants, in order to assess the effect of different energy saving measures.

The case study was carried out as part of the research project INFO¹, and parts of the work were also disseminated in various papers [12], [50], [51], [79] as well as in the final project report [71].

In the following sections, the process of design and model development for the case study is described (as well as some relevant background), to provide insights into the process of performing a co-simulation. However, it is to mention that some parts of the case study development process are not part of this thesis (for example developing the sub-models). These parts are only described briefly, but are necessary in order to present the overall context.

2.1 Background

The following section presents some relevant background and state of the art regarding co-simulation. Starting from an attempt to classify concepts related to coupled modeling and simulation in general, a definition of co-simulation is given. After that, common coupling strategies are presented and compared as well as relevant technologies and tools for implementing co-simulation solutions.

¹<http://projekt-info.org>

2.1.1 Terminology

As mentioned in the introduction, apart from 'classical' modeling and simulation in a single simulation environment, more and more approaches are now trying to couple multiple equation solvers and/or multiple simulation environments [17], [21], [32], [41], [47], [88], [131]. For continuous systems, [38] aims to classify coupling methods, resulting in table 2.1.

Table 2.1: Classification of methods for coupled simulation (adapted from [113] and [135])

	Monolithic simulation (single numerical solver)	Distributed simulation (multiple numerical solvers)
Monolithic modeling (single modeling environment)	I: Classical simulation	II: Model separation
Distributed Modeling (multiple modeling environments)	III: Model coupling	IV: Co-simulation

This table is also applicable for hybrid discrete/continuous systems if the term 'numerical solver' is interpreted more broadly to also include discrete-event schedulers.

Modeling and simulation using a single simulation tool, i.e. '*classical*' *simulation* (quadrant I) without coupling, is still the most present method. The simulation tool is often specialized for the particular application domain (or covers multiple domains, for example Modelica/Dymola), and provides libraries of pre-defined modeling elements that can be reused to build complex models in a time-efficient manner [135]. These elements have a mathematical or logical description with varying degree of detailing, depending on the application. After the overall model has been parametrized, it is usually compiled to arrive at an executable simulation, during which a numerical algorithm is used for calculation that is provided by the simulation tool and is usually tailored to work well in combination with the provided model description.

Besides classical simulation, it is often possible to implement individual model parts in different modeling environments and export these models for importing them and simulating them in a single simulation tool using a single solver. This method can be called *model coupling* (quadrant III) or strong coupling [131]. The model export can be carried out by exporting equations or simulation code, either as symbolic equations, source code or compiled code [113]. In some cases, discretized equations are exported, meaning that the numerical solver algorithm is also part of the export, which is why this method falls under co-simulation (quadrant IV) in the classification used here. One

prominent example in this regard is the Modelica Functional Mock-up Interface [13], see also section 2.1.3 for more details.

In contrast, if only a single simulation environment is used for model implementation, in which the model is then divided up into multiple simulation algorithms, this can be called *model separation* (quadrant II). This allows for example to parallelize the simulation or to separate stiff equations for solving them with an individual step size. This leads to so-called *multirate* methods where two or more different step sizes are used for distinct parts of the model [113], [123].

Finally, *co-simulation* (quadrant IV, sometimes also called solver coupling or simulator coupling) uses multiple modeling environments as well as multiple numerical solvers, allowing combining multiple step sizes as well as solver algorithms (e.g. explicit/implicit). This is also referred to as *multirate* or *multimethod* simulation. In many cases multidomain or multidisciplinary simulation models that use co-simulation are divided into monodisciplinary sub-models that are then implemented in specialized simulation environments [44], [113].

This classification provides a guideline on how to differentiate co-simulation from other methods. There are similar definitions in various publications, for example [2, pp. 27-28] defines co-simulation as an approach in which

...the subsystems are integrated by different time integration methods such that each of these methods can be tailored to the solution behaviour of the corresponding subsystem.

Other definitions are given for example in [87, p. 17], in which

...co-simulation is used to solve a coupled system by simulating each part with its own coupleable simulation tool

or in [86, pp. 93-94], which states that co-simulation

...is a rather general approach to the simulation of coupled technical systems and coupled physical phenomena in engineering with focus on instationary (time-dependent) problems

and that it exploits

...the modular structure of coupled problems in all stages of the simulation process beginning with the separate model setup and pre-processing for the individual subsystems in different simulation tools.

In [122, p. 249], these definitions are summarized from an engineering point of view for the co-simulation process as

...a simulation process of the whole system, where two or more subsystems are connected between each other in one simulation environment by specialised communication interface(s) with a pre-defined time step for data exchange.

As we concur with these definitions in principle, we want to expand the descriptions to also include variable communication intervals and hybrid systems, and propose the following definition, which we will use for the remainder of this work:

Co-simulation (cooperative simulation) is a method for simulating heterogeneous (continuous, discrete or hybrid) system models (typically instationary and time-dependent) by combining multiple sub-models and simulation algorithms (integrators, event schedulers, etc.) from different simulation environments, where the sub-models exchange data during runtime via specialized communication interfaces.

This definition incorporates middleware solutions for co-simulations (like BCVTB, see section 2.2.2) as well as model import/export (for example with FMI, see section 2.1.3) and various coupling strategies (see section 2.1.2) with fixed or variable communication intervals as well as iterative schemes.

Co-simulation bears some similarities to *Hardware-in-the-Loop* (HIL) and *Software-in-the-Loop* (SIL) methods where numerical solvers are replaced with different software or hardware. However, HIL and SIL applications typically have to satisfy much stricter real-time requirements [5], [62], [141], for example when developing embedded systems [54]. In this context, also the term *Model-in-the-Loop* (MIL) is sometimes used [99], [144].

Co-simulation presents three immediately visible advantages (see also [135] and [12]):

- *Modeling advantage:* A co-simulation model may span multiple physical domains with each domain sub-model being treated in a specially suited simulation environment. This includes not only use of specially tailored model descriptions (e.g. physical equations, data-based methods, discrete-event systems), but also user interfaces and environments with which the individual domain experts are already familiar with, ultimately resulting in accelerated model development.
- *Simulation efficiency advantage:* Each domain sub-system can employ different algorithms for numerical calculations (e.g. differential equations solvers, discrete-event schedulers, etc.), each of the tailored to the individual needs of the particular sub-model (in terms of solver method, step size, etc.), resulting in a more time-efficient co-simulation.
- *Engineering advantage:* Simultaneous model development in each engineering domain allows accelerating the engineering and development process.

On the other hand, prominent disadvantages of co-simulation include additional overhead and thus reduced performance due to the necessary data exchange as well as possible stability issues due to additional numerical errors [113], [135]. More details on this are given in section 2.1.2 and chapter 4.

2.1.2 Coupling Strategies

As soon as there are multiple coupled numerical solvers involved (i.e. model separation and co-simulation with regard to table 2.1), strategies are necessary to approximate exchanged data between communication points and to synchronize the simulators involved [113]. The typical coupling strategies for dynamic data exchange at runtime found in most literature are *loose coupling* (also called *weak coupling*) and *strong coupling* [3], [46], [129], [135]. While in loose coupling the simulators exchange data only at discrete points in time (between these points the sub-models are calculated independently), strong coupling implies that data is exchanged iteratively in each time step (in order to meet certain convergence criteria).

Within the loose coupling scheme, two different types can be differentiated, i.e. so-called *Jacobi type* and *Gauß-Seidel type*. While the Jacobi type describes parallel data exchange between the simulators, Gauß-Seidel type employs sequential (alternating) communication. Figure 2.1 presents an overview of the computation and data exchange procedures for the different coupling strategies.

In addition to loose and strong coupling, the *dynamic iteration* scheme presents a synthesis of both strategies by repeating a macro-step multiple times in order to improve convergence [114].

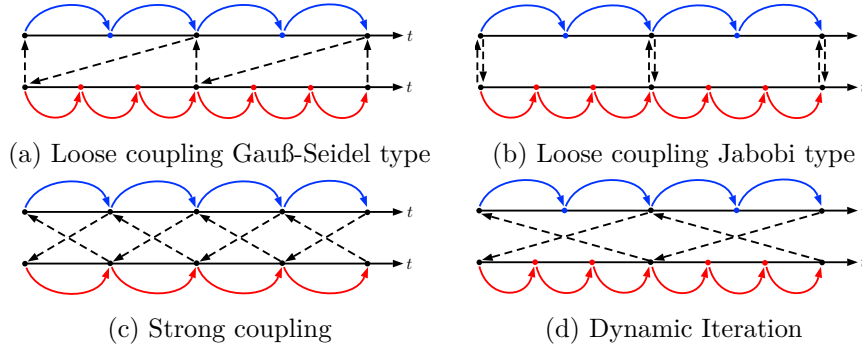


Figure 2.1: Different coupling strategies for co-simulation (adapted from [114] and [113]). The blue and red arrows denote computation steps in simulator 1 and simulator 2, respectively. The black arrows denotes data exchange.

In the relevant literature, these coupling strategies can also be found under different terms:

- *Loose coupling Jacobi type* [25], [46]: Weak coupling [3], ping-pong coupling [52], Quasi-dynamic coupling [143], Conventional Parallel Staggered (CPS) Procedure [30], naive modification for parallel processing [32].
- *Loose coupling Gauß-Seidel type* [46]: Conventional Serial Staggered (CSS) Procedure [30], sequential staggered solution [32].
- *Strong coupling* [25]: Fully-dynamic coupling [143], Monolithic approach [123].
- *Dynamic iteration* [114]: Waveform relaxation [29], [76], [83].

In some literature, the term weak coupling (especially Jacobi type) is even used synonymously with co-simulation (see for example [3, p. 26]), while others also use the term *solver coupling* [116].

For practical application, loose coupling presents the communication strategy that is easiest to implement since no iterations are necessary that would require the simulators to jump back in time (including re-initialization). The communication also does not require deep integration into the numerical solver (like strong coupling). This is why loose coupling (especially Jacobi type) is the strategy found most often in co-simulation solutions, especially with commercial tools [113].

This motivates to briefly discuss in the following Jacobi type and Gauß-Seidel type of loose coupling to illustrate their similarities and differences. For this, we consider a simple example of two systems of ordinary differential equations with states x_1 and x_2 , initial conditions $x_1(0) = x_{1,0}$ and $x_2(0) = x_{2,0}$ and coupled input/output $u_i, y_i, i \in \{1, 2\}$ [135]:

$$\text{System 1: } \dot{x}_1 = f_1(x_1, u_1), \quad y_1 = g_1(x_1, u_1), \quad (2.1)$$

$$\text{System 2: } \dot{x}_2 = f_2(x_2, u_2), \quad y_2 = g_2(x_2, u_2), \quad (2.2)$$

$$\text{Coupling: } u_1 = y_2 \text{ and } u_2 = y_1. \quad (2.3)$$

The values of y_1 and y_2 are the ones being exchanged between the two systems (i.e. the coupling) and usually depend on the internal states as well as other inputs. Synchronization and data exchange takes place at equidistant time steps $\{t_0, t_1, t_2, \dots, t_{N-1}, t_N\}$. Each of these two systems may be calculated using a separate numerical algorithm. In particular, let Φ_1 and Φ_2 denote the individual update functions that compute the values of the state variables x_1^{k+1} and x_2^{k+1} , respectively, at the next synchronization time step $t_k, k \in \{0, 1, 2, \dots, N\}$ (so-called *macro-steps*) together with respective outputs y_1, y_2 :

$$x_1^{k+1} = \Phi_1(x_1^k, \tilde{u}_1^k), \quad y_1^{k+1} = g_1(x_1^{k+1}, \tilde{u}_1^k), \quad (2.4)$$

$$x_2^{k+1} = \Phi_2(x_2^k, \tilde{u}_2^k), \quad y_2^{k+1} = g_2(x_2^{k+1}, \tilde{u}_2^k). \quad (2.5)$$

The necessary input values $\tilde{u}_1^k, \tilde{u}_2^k$ have to be extrapolated or interpolated (depending on the coupling scheme) from u_i^k and $u_i^{k+1}, i \in \{1, 2\}$, respectively [46]. See the following sections on Jacobi type and Gauß-Seidel type for more details.

Note that the update functions Φ_1 and Φ_2 are defined by a sequence of code instructions executed in the respective simulation engine. For continuous systems like in equation (2.1), these update functions typically represent differential equation solvers, For more general systems (i.e. discrete, hybrid), they can incorporate arbitrary logic.

During computation of the respective update functions, each simulator may perform several independent steps at discrete times $t_\xi \in [t_k, t_{k+1}]$ (in this context they are also called *micro-steps*), in order to advance from t_k to t_{k+1} . However, these steps are not communicated to the outside.

Jacobi Type

For the Jacobi type of loose coupling, each simulator can be executed in (quasi-) parallel. To advance from time t_k to t_{k+1} , each simulator used its own update function Φ_1 and Φ_2 , respectively (see equation (2.4)), for which the input values have to be extrapolated using u_1^k resp. u_2^k :

$$\tilde{u}_1^k(t) = h_1(t, u_1^k), \quad (2.6)$$

$$\tilde{u}_2^k(t) = h_2(t, u_2^k), \quad (2.7)$$

with extrapolation functions h_1 and h_2 . This is because the values for u_1^{k+1} and u_2^{k+1} , respectively, are available only after the next communication at time t_{k+1} . In many cases, this extrapolation is simply done by using the constant value, i.e. $\tilde{u}_1^k(t) = u_1^k$, $\tilde{u}_2^k(t) = u_2^k$, but also higher-order extrapolations are possible.

At the end of the time step, the simulators exchange their new outputs y_1^{k+1} respectively y_2^{k+1} with one another (see also the coupling in equation (2.1)):

$$u_1^{k+1} = y_2^{k+1}, \quad (2.8)$$

$$u_2^{k+1} = y_1^{k+1}. \quad (2.9)$$

Figure 2.2 shows the periodic sequence for data exchange between two coupled simulators.

Gauß-Seidel Type

When using Gauß-Seidel type of loose coupling, the simulators are executed sequentially during each communication interval $[t_k, t_{k+1}]$. The update function that is executed first – let that be Φ_1 without limitation of generality – uses again an extrapolation of its input u_1^k :

$$\tilde{u}_1^k(t) = h_1(t, u_1^k). \quad (2.10)$$

At time t_{k+1} the new output value $y_1^{k+1} = u_2^{k+1}$ is communicated to simulator 2, which can then use an interpolation for its input,

$$\tilde{u}_2^k(t) = \tilde{h}_2(t, u_2^k, u_2^{k+1}), \quad (2.11)$$

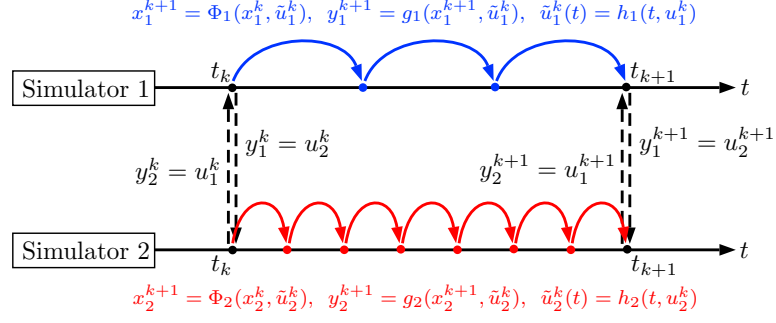


Figure 2.2: Data exchange between two coupled simulators according to Jacobi type of loose coupling. Both input variables \tilde{u}_1^k and \tilde{u}_2^k have to be extrapolated between communication intervals.

with an interpolation function \tilde{h}_2 . When simulator 2 is finished at time t_{k+1} , it communicates its own output value $y_2^{k+1} = u_1^{k+1}$ back to simulator 1. This periodic sequence is illustrated in figure 2.3.

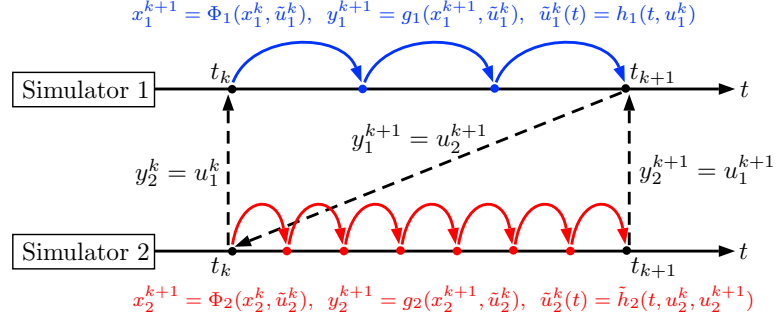


Figure 2.3: Data exchange between two coupled simulators according to Gauß-Seidel type of loose coupling. Simulator 1 is executed first and has to extrapolate its input \tilde{u}_1^k , simulator 2 can use interpolation for \tilde{u}_2^k .

Which simulator is executed first is usually chosen depending on system characteristics of the sub-model. In the *fastest-first* approach the stiff system having the higher frequency dynamics is executed first, while the *slowest-first* computes the system with lower frequency dynamics first [135].

The fastest-first approach promises smaller extrapolation errors (since the slower changing models are the ones being extrapolated) while using slowest-first has a lower risk of having to re-calculate (micro-) steps [123] and allows larger communication intervals [46].

To summarize, loose coupling presents one of the simplest forms of coupling schemes for co-simulation, especially when using fixed communication intervals. While Gauß-Seidel type reduces extrapolation errors during co-simulation, Jacobi-type allows parallel

execution of the simulators, a simpler implementation (e.g. using constant extrapolation) and the behaviour of the iterations does not depend on the calling sequence of the clients [10].

If discrete-event systems or state events are involved in one or more of the sub-models where the events influence other sub-models across the co-simulation, the coupling strategies have to become more sophisticated, because these events have to be communicated immediately and cannot wait until the next communication interval. For more details on this, we refer to relevant literature, e.g. [3], [16], [34], [40], [129].

2.1.3 Technologies and Tools for Co-simulation

High-level Architecture

The Modeling and Simulation High Level Architecture (HLA), culminated in the IEEE standard 1516-2010 [61], specifies a general-purpose architecture for interoperability of distributed simulation. For interaction between simulation systems (so-called Federates), it utilizes a runtime infrastructure (RTI), including a usually centralized data management middleware and a Federation Object Model (FOM) that specifies semantics of the data in a simulation.

Functional Mock-up Interface

Originating from the automotive industry, the Functional Mock-up Interface (FMI) [13], [86] is a different tool-independent standard with the goal to support the exchange of simulation models and co-simulation of dynamic models. In fact, the FMI specification distinguishes between *FMI for model exchange* and *FMI for co-simulation*. It is interesting to note that this distinction can also be explained with reference to section 2.1.2, where FMI for model exchange corresponds to a strong coupling strategy while FMI for co-simulation employs weak coupling (see also [3, p. 26]). But FMI for co-simulation also supports more sophisticated strategies, e.g. communication step size control or higher order signal extrapolation. While the FMI specification is designed to support a very general class of middleware (master) algorithms, it does not define the master algorithm itself [87].

FMI has gained popularity across different simulation tools, however it is focused predominantly on continuous models based on the Modelica language [37]. Although the Modelica language allows to incorporate hybrid model characteristics in principle (in terms of state events), it is not well-suited for event-driven simulations. For numerical calculations, Modelica restricts the ability to combine ODE solvers with discrete-event schedulers, resulting in workarounds and reduced performance. In this regard, there have been some investigations to overcome these limitations [82], [94]. Other research efforts are attempting to combine process-oriented modeling with Modelica, e.g. in the Modelica DESLib library [110], however these solutions are still lacking maturity for practical use.

Middleware solutions

There exists a multitude of software solutions for middleware-driven co-simulation, where custom software handles orchestration and coordination between the simulation tools [21], [39], [47], [88], [92], [106]. However, many of them are highly customized for specific simulation tools and/or application areas [115] with low reusability of model parts. Only a few middleware solutions, facilitate general-purpose use by including domain-independent scientific computing environments and multi-domain simulation tools. One of these tools is the Building Controls Virtual Test Bed (BCVTB) [139], which we use for the case study presented in this chapter. More details on BCVTB are given in section 2.2.2.

Nearly all of these middleware tools offer coupling and data exchange only on implementation level, meaning that only raw data is exchanged without inherent semantics. Therefore, managing and maintaining models and connections remains a challenge, which we will see in the following sections.

2.2 Design

2.2.1 Reference Model

In order to promote systematic model development and a coordinated cooperation between experts from different disciplines, a first step involves formalizing and documenting the structure of the overall system under consideration [78]. This component-based reference model is a generic description of a production plant with a focus on energy flows. It is intended to give participating developers an overview of the components the system is comprised of and, in particular, their dependencies and interfaces. This improves understanding of the overall system and can serve as a basis for a concrete implementation of individual model components, corresponding interfaces for communicating relevant state variables (defined in the reference model) and their subsequent coupling to a co-simulation.

Developing the reference model was not part of this thesis, but was developed as part of a larger research project. However, as described, it was used as a basis for the implementation of the co-simulation, which is why the reference model is briefly described below. For a more detailed description, we refer to the corresponding documentation [71] and [79].

Figure 2.4 shows the reference model of a production facility as a network of 16 parts with dynamic variable connections (black arrows) as well as parameter dependencies (green arrows). The model structure is based on a general description of components, parameters and variables. Each of the components represents a distinct part of the overall system and is found in a similar form in almost every production plant. A component does not necessarily have to be of a physical nature. More concrete, two types of components are distinguished:

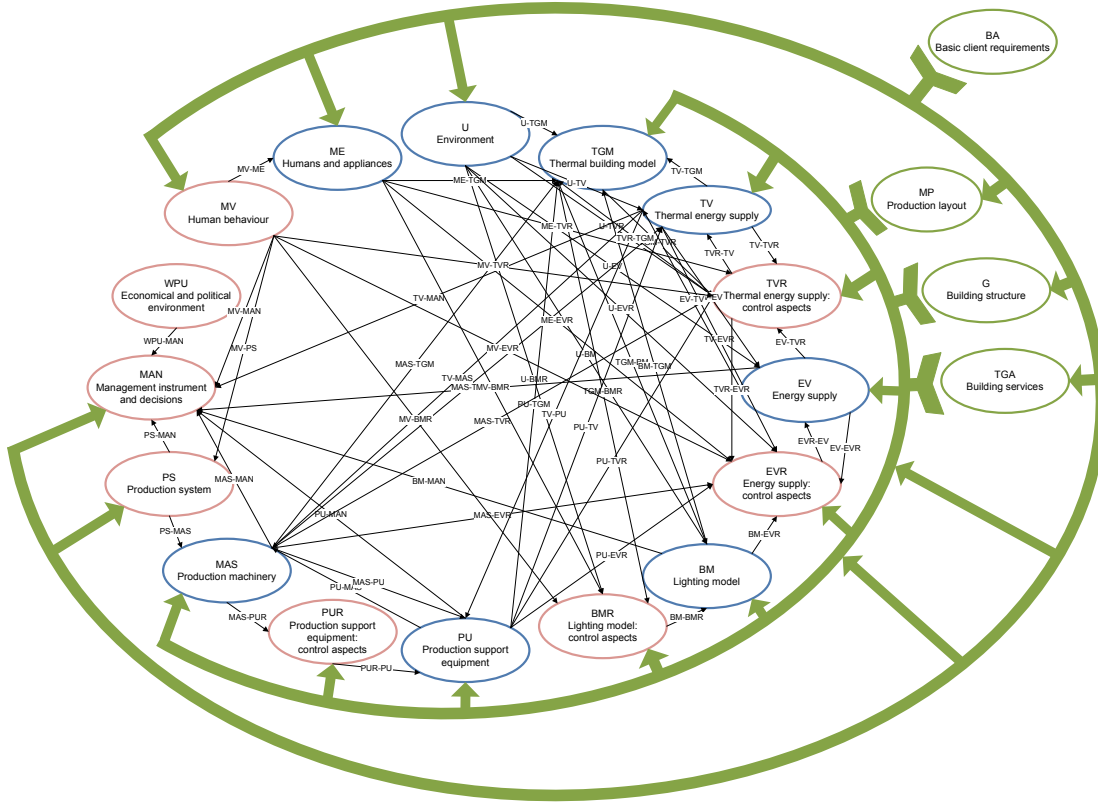


Figure 2.4: Reference model overview (taken from [71]). Physical components are in blue, information components in red, plan elements in green. Black arrows show dynamic variable connections, green arrows are static parameter associations.

- Physical components: Building structure, machine tools, people, the environment, etc.
- Information components: Control strategies and algorithms, behavioural models, political objectives, etc.

For encapsulating an individual component against the overall system, generic interfaces are specified, via which the components exchange information (in the form of variables) and thus interact with one another.

In addition to dynamic information exchange, planning components and parameter references (see figure 2.4) take into account static dependencies between components, that may occur for example during planning. These planning components are themselves not part of a dynamic simulation, but rather provide additional guidance during parametrization.

This generic reference model aims at providing a system overview and itself does not describe a concrete implementation of the internal behaviour of individual model elements.

Instead, it follows a black-box view of its components, thereby remaining independent of any concrete implementation language or simulation environment. This allows maximum flexibility for adapting individual parts to project-specific implementations and requirements, in terms of model complexity, data availability, or numerical algorithms.

For a concrete model instantiation, individual or groups of components can be modelled using physical relationships, parameter dependencies, data-based models or other rules, thus determining their internal behaviour in terms of continuous as well as discrete-time dynamics. Internal variables describe the current state of a component. Such variables may represent physical data points (heat quantity, temperature, etc.) as well as information states (e.g. status of a switch). Via the interfaces and dynamic variable connections defined in the reference model, these states also influence other components, creating dynamic dependencies, feedbacks and other complex interactions. For example, production machines can supply usable waste heat and electrical power demand to the building service model. Alongside, the thermal building model with its internal loads calculates room temperatures, air exchange as well as energy demand for room conditioning.

2.2.2 Co-simulation Architecture

Based on the reference model described in the previous section, the task was to implement a dynamic simulation of a production plant case study. In order to allow taking into account dynamic interaction across the different physical domains, data exchange between components is necessary at runtime – a mere ‘static’ model coupling in terms of sequential one-time simulation of each component would not be sufficient. An efficient implementation of a simulation model spanning several engineering domains (machines, energy system, building) also requires combining multiple model description formalisms (e.g. differential equations, state machines or data-driven modeling). Since there does not seem to be any suitable single simulation tool available that would have met all of these requirements and would have been able to fully model the overall system in the necessary complexity and level of detail, a approach was chosen using co-simulation between multiple simulation tools.

This approach also has a positive effect on the model development process, because each sub-model covered by a single simulation tool can be developed and verified independently. These models then have just to be extended to include necessary interfaces – which can be derived from the reference model – before coupling them to perform a co-simulation.

Simulator Clients

For implementing individual sub-models, we consider various tools for modeling and simulation of physical systems:

- MATLAB [126]: MATLAB (Matrix Laboratory) is a proprietary scientific numerical computing environment developed by MathWorks. It offers multi-paradigm

modeling, algorithms for numerical analysis (including ODE solvers), plotting and visualisation of data and creating user interfaces. Available additional packages include symbolic computing, Simulink for graphical multi-domain dynamic simulation, Simscape for modeling and simulation of physical systems among a number of others².

- Dymola [23]: Dymola (Dynamic Modeling Laboratory) is a modeling and simulation environment for component-based equation-oriented modeling and simulation based on the Modelica modeling language [85] (see also section 2.1.3). It allows simulating physical-technical systems of multiple engineering domains, such as mechanical, electrical, thermodynamics, hydraulic among others. Additional model libraries (commercial as well as open-source) provide extended components for specialized applications (powertrains, vehicle dynamics, air conditioning, etc.)³.
- EnergyPlus [130]: EnergyPlus is simulation tool for energy simulation of whole buildings. It allows to model energy consumption for heating, cooling, ventilation, lighting as well as other loads and calculate thermal zone conditions, heat and mass transfer, and illumination. It can also take into account ambient weather conditions. EnergyPlus is funded by the U.S. Department of Energy Building Technologies Office. The core software is open-source and cross-platform with various additionally available graphical front-ends⁴.

Middleware

The communication between different simulation environments has to be managed by some additional software, so-called middleware, illustrated in figure 2.5. This middleware not only coordinates the data exchange at runtime, but also ensures the synchronization between the individual simulators.

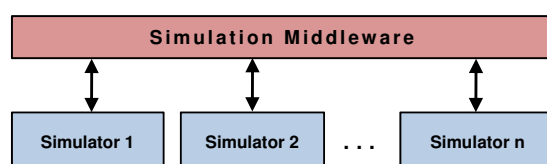


Figure 2.5: Communication and data exchange between simulation tools is managed via middleware.

For the implementation of a co-simulation between the simulation tools mentioned above, a prototypical open-source software framework, called BCVTB (Building Controls Virtual

²See also <https://www.mathworks.com/products/>

³For an overview of available libraries, see <https://www.modelica.org/libraries>

⁴See also <https://energyplus.net/interfaces>

Test Bed) [140], is available. BCVTB is developed at the Lawrence Berkeley National Laboratory at the University of California [139]. Although BCVTB was initially intended to be used in the area of building simulation, it can also be used for co-simulation in other application fields, since it supports several multi-domain and domain-independent simulation tools.

The BCVTB software follows a client/server architecture, illustrated in figure 2.6 for two clients, where during initialization the BCVTB server calls the simulator clients ("System Call") and passes them a configuration file used to set up the communication. The actual data exchange during runtime is carried out between the client and the BCVTB server via a BSD socket interface [140] for interprocess-communication using the TCP/IP internet protocol family, which enables the co-simulation to be run over a computer network [120]. A BCVTB director is responsible for passing data received from one client on to another client according to a routing topology specified by the user.

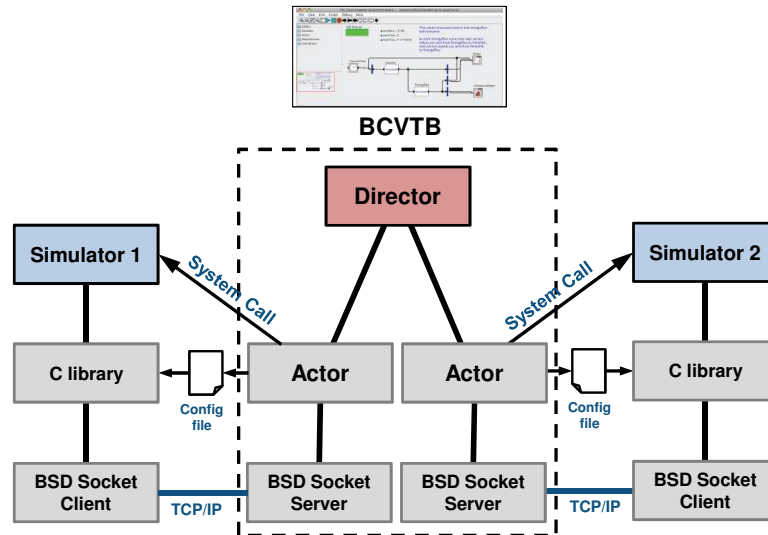


Figure 2.6: Architecture overview of the BCVTB software (adapted from [139]).

Overall Framework

Following figure 2.5, figure 2.7 shows an overview of the implemented co-simulation framework with the software tools involved. For managing and processing the simulation results, a second instance of MATLAB was provided. This instance is also a central point of contact for the user to execute the simulation as well as to visualize the results, provided via a graphical user interface in MATLAB, shown in figure 2.8.

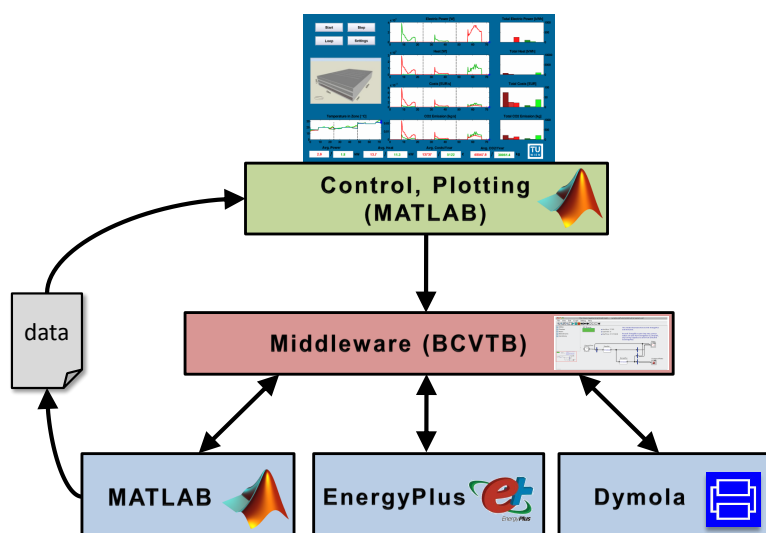


Figure 2.7: Overall framework for co-simulation between MATLAB, Dymola and EnergyPlus. A second MATLAB instance provides processing and visualization of simulation results.

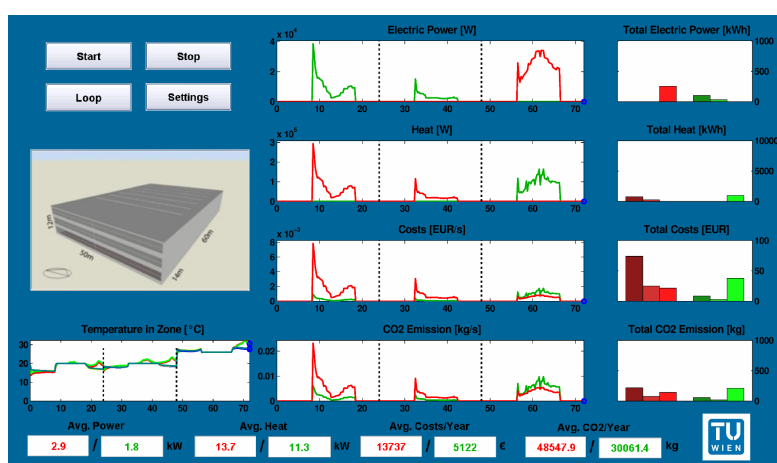


Figure 2.8: Graphical user interface in MATLAB for executing the simulation and plotting simulation results.

For logging and export of simulation results, the intuitive approach would be to tap into the BCVTB server and from there pass the data to the second MATLAB instance. However, due to usability issues with BCVTB, it was easier in the present implementation to instead transfer the data to one of the clients, because it allowed for a much simpler implementation of data processing and coordinated read/write access for a 'live preview' of the results data in the graphical user interface (cf. figure 2.8) [71].

2.2.3 Data Exchange and Synchronization

BCVTB employs data exchange between the different clients using a fixed synchronization time step without iteration [140]. With reference to section 2.1.2, this corresponds to a loose coupling scheme of Jacobi type. Some restrictions result from this coupling scheme:

- According to the Jacobi type scheme, inputs have to be extrapolated between communication intervals, leading to increased numerical errors and in the worst case even to stability problems.
- Although BCVTB allows to couple discrete and continuous models in theory, data exchange is limited to fixed communication intervals, meaning that discrete events cannot be communicated immediately to the other sub-models (like with other coupling schemes, see [40], [129] for example). This not only increases numerical errors but may even lead to unintended behaviour, which makes BCVTB not well suited to include event-driven sub-models.

In addition, some care has to be taken regarding communication at the start of the simulation, where initial values have to be exchanged including synchronization in such a way that it does not matter which of the simulators is called first [140].

2.3 Implementation

Based on the co-simulation framework design described in the last section, a concrete instance of a coupled simulation of a production facility case study could be implemented. The developed reference model (see section 2.2.1) serves as the starting point for a coordinated development of concrete implementations of the individual sub-models as well as instantiating the co-simulation middleware. Developing and implementing the sub-models themselves was not part of this work, but instead was carried out by several domain experts within the scope of the research project INFO [71], which is why this part is not described in detail here. The sub-models are however presented briefly, as some insights are necessary for understanding of the overall co-simulation case study. For a more comprehensive description, refer to [71]

2.3.1 Sub-Models

In coordination with the co-simulation framework design, the following sub-models were implemented in the respective simulation environments:

- Machines and production system: MATLAB/Excel
- Energy system, building services and control aspects: Dymola
- Thermal building model and lighting: EnergyPlus

may cause numerical problems. These can be caused on the one hand by a large number of coupled equations which have to be solved iteratively, and on the other hand also by widely differing time constants between the components leading to stiff systems of differential equations. See also [48] for more details.

Also, for simulating larger time periods (e.g. one week up to an entire year) as part of a co-simulation together with building and energy system models, it is strictly necessary to reduce the complexity of machine models by only incorporating main energy flows relevant to the particular investigation. For this, alternative modeling approaches have to be considered, for example using data-driven parametric models with significantly lower temporal resolution (e.g. minutes instead of sub-seconds) and identifying energetically relevant parameters from higher-resolution measurement data as well as from results of detailed simulation models (e.g. like the one described above). From this high-resolution data, different recurring operating states can be identified together with corresponding energy levels and averaged durations, resulting in discretized load profiles.

For example, power consumptions can be compared in idle mode, during chipping and without chipping, from which three basic energy levels can be identified (see also figure 2.10): The base load P_{base} , the dynamic load P_{dyn} (e.g. from drive motors) and the processing-related load P_{cut} (for actually cutting the material). Averaged temporal shares of individual energy levels on the overall processing time can be determined from detailed simulations of production processes and analysing load profiles.

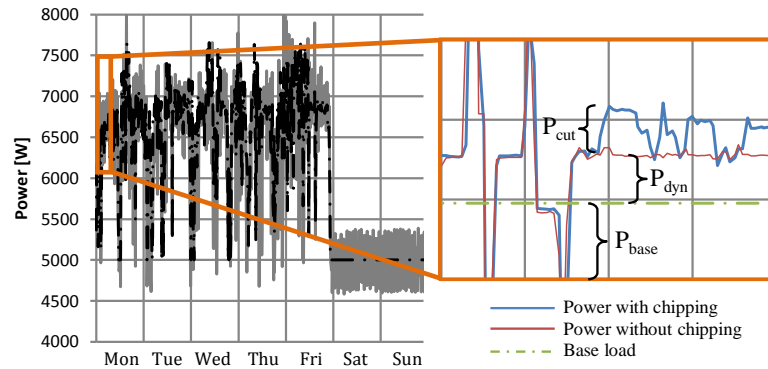


Figure 2.10: Power consumption of a machine tool over time (left) with detail (right) showing base load P_{base} , dynamic load P_{dyn} and cutting power P_{cut} (adapted from [50]).

In addition to the electrical power consumption, its conversion to thermal waste heat must also be studied. Some of this was already presented in [72]. The analysis shows that only a small fraction (in most cases less than 1 %) of the total energy demand ends up stored in the material itself by machining operations. The remaining energy used to power electrical drives is subsequently converted to waste heat due to friction and other

power losses. This waste heat is transferred to the environment or can be reused via heat recovery systems. See also [71] for more details.

The described steps were used to instantiate the parametric machine tool model by specifying⁵ parameters (specifically energy levels and averaged time durations) for an existing machine pool in the scope of the case study. These instances were implemented in Excel⁶ in combination with MATLAB. The model computes the energy demand, as well as reusable and waste heat to the surrounding area. In the overall co-simulation, this waste heat is transferred to the building model (as heat gain), reusable heat and energy demand serve as input for the energy system model, respectively [50].

From a modeling perspective, it is important to note that this form of simplified load profile based on states and time durations results in a purely discrete (time-driven) model, as oppose to the continuous model shown in figure 2.9.

For validation, comparison of simulated load profiles and aggregated energy consumptions with additional measurement data show excellent agreement with deviations in daily energy consumption of less than 3 % [12], [71]. This not only confirms the feasibility of the simplified modeling approach, which, as mentioned, is not only possible but necessary for an overall co-simulation together with other sub-models, and that the resulting deviation is practically negligible.

Building

The building sub-model describes the thermal aspects of the building hull and thermal zones, which house the production. The model includes time schedules for heating, cooling, occupancy and lighting as well as environmental influences (ambient temperature, weather conditions, etc.). It does not include any building services since they are part of the energy system model. During simulation, the model processes heat gains from people, lighting (and possibly heat gains from production machines injected from the co-simulation) and calculated heating and cooling demands.

For composing the building model, Building Information Modeling (BIM) methodologies and tools were applied. BIM provides a parametric, three-dimensional digital building model, but moreover a methodology to manage the essential building design and project data in digital format throughout the life cycle of a building [28]. The building geometry, floor layout, load bearing structure and hull are assembled and then exported to the thermal building simulation tool EnergyPlus in order to perform the simulation.

⁵Going beyond that, it would even be possible in theory to dynamically adapt these parameter values during simulation runtime by periodically invoking a detailed simulation for a short period of time, e.g. for a one-time simulation of a repetitive production process [12]. This however has not been studied in the current work and did not seem necessary for the level of abstraction used in the co-simulation framework.

⁶This was due to practical advantages in the course of the project.

Energy System

The energy system model describes all technical building services (TBS), i.e. equipment for supplying the production plant with electric and thermal energy, e.g. pumps, chillers, photovoltaic system and heat pumps. The model receives energy demand for production and building conditioning as input and calculates primary and final energy demand. For comparing design variants of the energy system, three different models have been implemented using the Modelica modeling language and Dymola simulation environment, see section 2.4 for more details.

Bearing in mind the intention to conduct a co-simulation of the energy system model together with other sub-models, special requirements regarding simulation performance and numerical stability had to be considered that differ from an isolated simulation. Because of this, a classical signal-flow-based modeling paradigm was employed instead of the acausal equation-oriented approach typically associated with Modelica [37]. This reduced description avoids overhead from (for this application unnecessarily) high modeling details (e.g. two state variables to describe energy flow), yielding less model equations, higher simulation speed and improved numerical stability, not least because of avoiding high-index differential-algebraic equation systems and their index reduction [18]. For instance, in [36] the authors describe numerical complications when using conventional Modelica fluid connectors. For validation of model components, simulation results were compared with data from test certificates and common literature of sub-systems like pumps, chiller and heat pumps.

2.3.2 BCVTB Middleware

Using the BCVTB software tool described, the middleware was instantiated and the interconnection network was implemented.

The necessary data exchange including its semantics could be derived from the reference model (see section 2.2.1). The implementation of this schema is shown in figure 2.11.

BCVTB provides components, called Actors, that handle the low-level data exchange with the individual clients during runtime, including establishing the socket connection (see figure 2.6). These actors are coordinated by a superordinate so-called director, that triggers the actors and determines the communication step size among other things. BCVTB also provides a graphical user interface for instantiating and managing the Actors as well as implementing the interconnection network by drawing signal connections, which can be helpful especially for non-expert users. The graphical user interface in BCVTB for the current case with Actors and signal connections is shown in figure 2.11.

On the client side, the sub-models have to be configured and extended by the respective interface connections for data exchange. This can be achieved in the following way:

between individual sub-groups of simulator clients with individual communication step sizes.

As mentioned, the actual data exchange between the BCVTB server and the simulator clients is achieved via socket connections (see also figure 2.6). As this is a rather low-level form of handling data, variables have to be prepared and combined into a single data vector before it can be transmitted over the socket. In BCVTB, these vectors have to be manually divided up into its components before assembling them into new vectors for the other clients according to the interconnection network. This procedure can also be seen in figure 2.11. This process is not only cumbersome for larger models but also very error-prone, especially if multiple people are involved. It is very important that the persons instantiating the client and server interface must therefore agree on a common semantics of the data vectors (i.e. which vector component represents which value, in which unit, etc.). This semantics is not per se given by the reference model because it represents an aspects that depends on the implementation and the participating simulators.

Finally, the implementation of the overall framework is presented in figure 2.12. As mentioned, the MATLAB client exports the simulation results, which can then be used for post-processing, e.g. by Excel or again in MATLAB.

2.3.3 Validation

Initial simulation results from the case study underwent detailed plausibility checks and were validated by independent calculations. These are not presented in detail here, for more we refer to [71]. Unfortunately, real-world validation was restricted due to lack of real measurement data.

2.4 Testing and Results

The implemented and validated framework instance was then used to simulate different scenarios and parameter settings and to analyse the impact of various energy saving measures on the overall system [50]. In addition, well-defined interfaces allow replacing individual sub-models with different instances without affecting other sub-models or the framework, in order to compare for example different configurations for the energy system and evaluate these variants with regard to energy efficiency, CO₂ emissions or economic viability.

In the following, exemplary simulation results are presented from the case study at hand in different scenarios, in order to demonstrate the applicability of the dynamic co-simulation and also to show which kind of conclusions are possible from this simulation-based analysis. A more comprehensive description of results is also presented in [71] or [50].

The real-world case study focuses on an industrial production facility for a high-end metal-cutting company performing small series production, with approximately 500 employees, 48 machine tools (machining centers, ovens and laser cutting machines) and

an annual energy consumption of close to 8 Mio. kWh [71]. The new production building with 20500 m² production space and 15000 m² office space and representative areas was designed for energy efficiency and flexibility [72].

For the energy system model, three different design variants are compared in order to find the configuration most suitable for the production facility in terms of energy efficiency. The three variants differ mainly in the type of energy supply, see table 2.2 for an overview.

Table 2.2: Overview of the simulation scenarios comparing different design variants for the energy system

	Scenario 1	Scenario 2	Scenario 3
Heat supply (base load)	combined heat and power (natural gas)	district heat	groundwater well & heat pump
Heat supply (peak load)	district heat	district heat	district heat
Heat recov. from exhaust air	heat pump	heat exchanger	heat pump
Warm water supply	solar thermal & electric	solar thermal & electric	solar thermal & electric
Cold supply	absorption chiller	compression chiller	groundwater well
Electricity supply	photovoltaic, combined heat and power & grid	photovoltaic & grid	photovoltaic & grid

Simulation results compare the final energy demand for the three scenarios over a simulation of one year, depicted in figure 2.13. The co-simulation requires a fixed communication step size, for which 15 minutes was chosen.

Scenario 1 presents a significantly higher energy demand compared to the other scenarios, which seems counter-intuitive at first because the scenario – just like the other ones – was designed using state-of-the-art knowledge regarding energy efficiency technologies.

In order to uncover the reason behind this seemingly inefficient scenario 1, further investigations are necessary. When looking at a more detailed breakdown of the energy flows through the system, as presented in the Sankey diagram in figure 2.14, it becomes obvious that a significant energy demand is caused by inefficient operation of the absorption chillers that convert heat into cooling energy. It turns out that these absorption chillers had been designed (especially regarding choice of nominal power) to be able to cover peak

cooling demands, that they however were operated only at partial load most of the time (there was also no cold storage available), where these machines have very poor efficiency [95]. This results in a high gas demand for the combined heat and power (CHP) unit (see also table 2.2) that supplies the absorption chillers with heating energy. As a by-product of the CHP, more electric energy is produced that can be fed back into the energy grid.

For a detailed energy flow analysis of scenarios 2 and 3 as well as further evaluation of simulation results (e.g. life-cycle cost-benefit analysis), we refer to [71].



Figure 2.12: Overall co-simulation framework instance for the case study with BCVTB middleware as well as MATLAB, Dymola and EnergyPlus simulation clients and post-processing in MATLAB.

2. CO-SIMULATION CASE STUDY

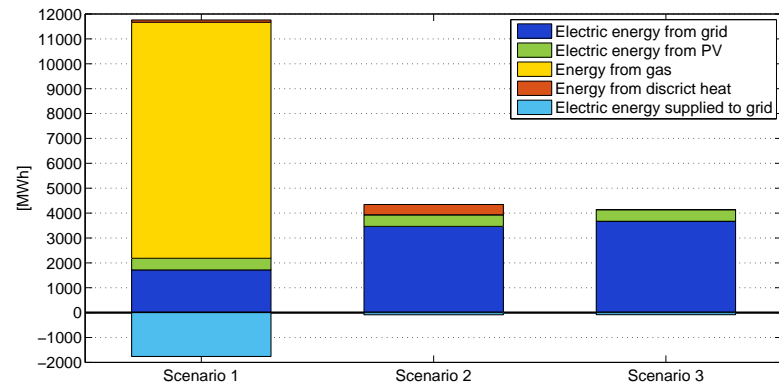


Figure 2.13: Comparison of annual final energy demand for the three scenarios. Scenario 1 shows demand for natural gas to operate the CHP unit.

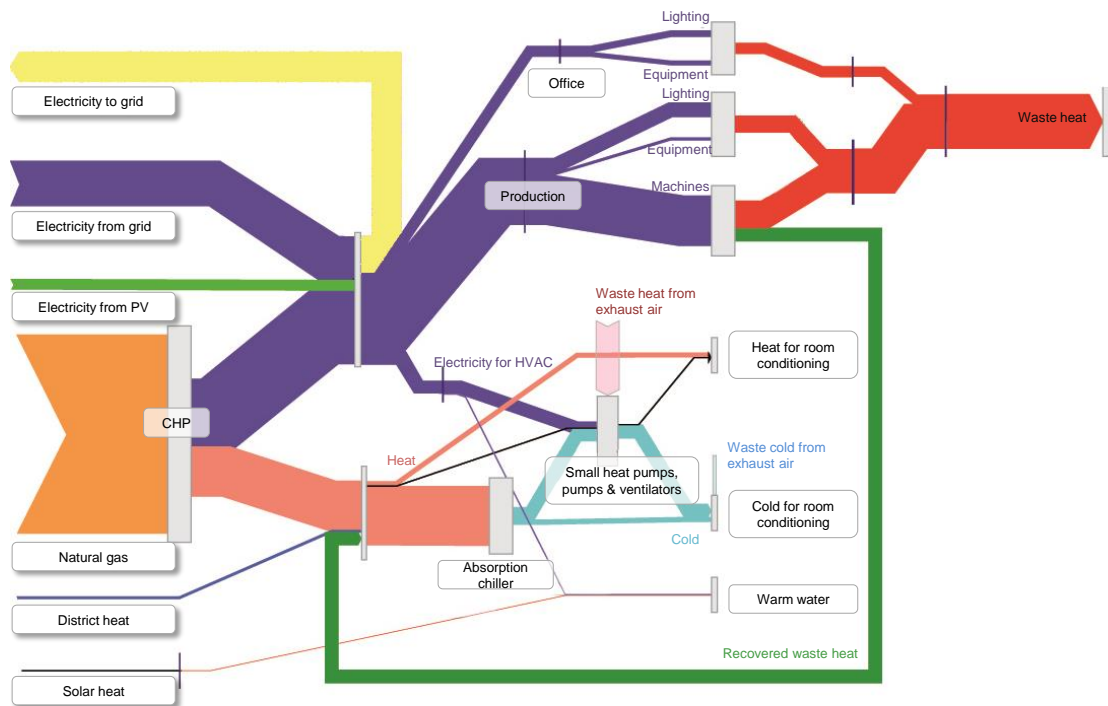


Figure 2.14: Detailed energy flow for the energy system configuration of scenario 1 (adapted from [71]). The analysis shows that the high gas demand is a results from inefficient operation of the absorption chillers.

DEVS-based Modeling Case Study

3.1 Background

This section presents some relevant background regarding the modeling formalism and tools, on which the case study implementation is based.

3.1.1 DEVS-based System Specification Formalisms

The first DEVS formalism was introduced by Zeigler in 1976 [142] as a formalized description of discrete-event systems, alongside similar specifications for discrete-time systems (DTSS) and continuous differential equation systems (DESS).

Starting from an introduction into the foundational Classic DEVS formalism, we present in the following extensions towards hybrid systems. All these DEVS-based formalisms have in common that they provide means to build models from components. In particular, they distinguish between *atomic* and *coupled* components. While atomic components constitute the main building blocks modeling a specific internal behaviour, coupled components are connections of blocks (either atomic or again coupled blocks) into larger models. In addition, most DEVS-based formalisms provide an important property, called *closure under coupling*, that guarantees that a coupling of systems in a DEVS formalism defines a basic system in the same formalism. In other words, a coupled component always behaves like an equivalent atomic component when looked at from the outside [66]. Closure under coupling allows to use networks of components as components in a larger coupled system. Together, this leads to a hierarchical, modular construction of system models.

Classic DEVS

The Discrete Event System Specification (DEVS) describes models with dynamics that allow changes only at discrete points in time, called *events*. An *atomic* DEVS model is defined by the tuple (see [142, p. 138])

$$DEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta) \quad (3.1)$$

where

- X is the set of input event values,
- Y is the set of output event values,
- S is the set of state values,
- $\delta_{ext} : Q \times X \rightarrow S$ is the *external state transition function*,
- $\delta_{int} : S \rightarrow S$ is the *internal state transition function*,
- $\lambda : S \rightarrow Y$ is the *output function*,
- $ta : S \rightarrow \mathbb{R}_0^+ \cup \infty$ is the *time advance function*,
- $Q = \{(s, e) | s \in S, e \in [0, ta(s)]\}$ is the set of *total states*.

To be precise, X , i.e the set of input event values, is the set of all possible values that an input event can adopt [69]. So for example X may be \mathbb{R}^n for n input ports. The same goes of course for Y . The functions δ_{int} , δ_{ext} , λ and ta define the system dynamics. In short, the δ_{ext} function is executed upon an incoming (external) event, δ_{int} reacts to an internal event and λ computes the output values. An internal event itself is triggered when time duration $ta(s)$ is exceeded. So $ta(s)$ is a (non-negative real) number that specifies how long the system remains in a given state in absence of input events.

A more detailed description of the internal behaviour and the execution sequence is given in the following, since it deems essential to understand in order to be able to implement models using the DEVS formalism. Figure 3.1 shows the behaviour on an atomic DEVS and figure 3.2 illustrates an example sequence of input and output events with corresponding state trajectory.

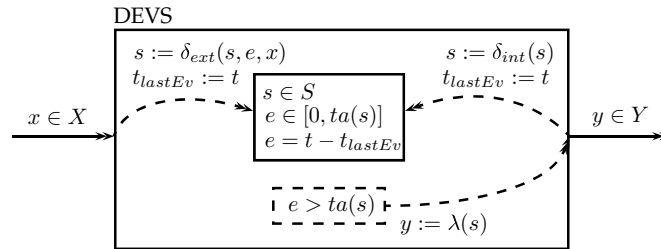


Figure 3.1: Operating principle of an atomic DEVS (taken from [102]).

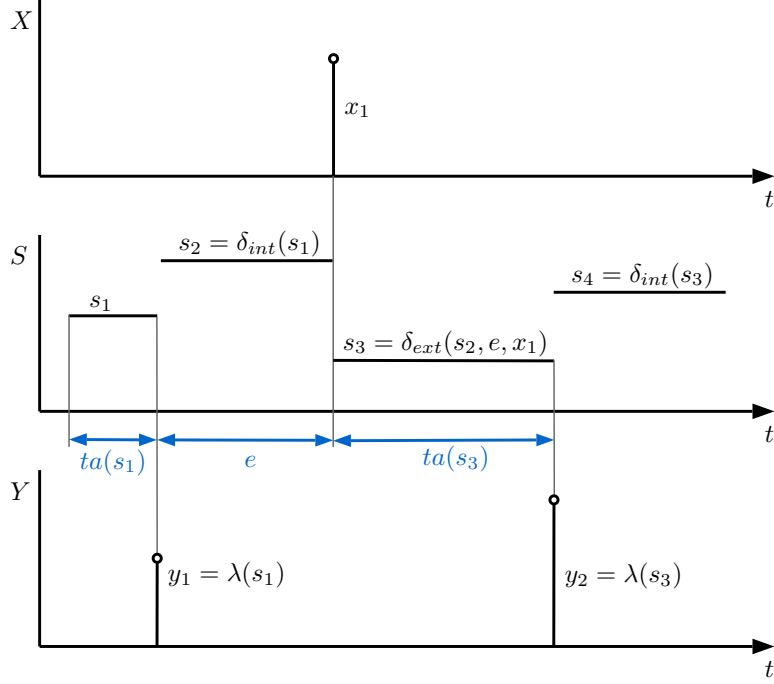


Figure 3.2: Input/Output events and state trajectory for an atomic DEVS model (adapted from [69]).

After the system state has adopted the value s_1 at time t_1 , the system waits for $ta(s_1)$ units of time, after which it performs an internal transition at time $t_1 + ta(s_1)$, changing the state to $s_2 = \delta_{int}(s_1)$, but only after triggering an output event with the value $y_1 = \lambda(s_1)$ ¹. When an external event arrives at time $t_2 + e$ (assuming $ta(s_2) > e$), the state changes instantaneously to $s_3 = \delta_{ext}(s_2, e, x_1)$ using the external transition function δ_{ext} with respect to the input x_1 , the previous state value s_2 and the elapsed time e since the last transition. Afterwards, the elapsed time e is reset to zero. No output is produced during an external transition [69], [102].

There are some restrictions for modeling atomic DEVS components, most notably *legitimacy* of atomic DEVS, meaning that for each possible set of initial conditions, only a finite number of events may occur during a finite amount of time. More precise, a DEVS is defined as legitimate [142, p. 142] if for each $s \in S$,

$$\lim_{n \rightarrow \infty} \sum (s, n) \rightarrow \infty \quad (3.2)$$

In addition to atomic DEVS, the formalism also specifies *coupled* DEVS, which are comprised of an external interface (input/output ports and values), sub-components

¹Note that y_1 may also be \emptyset , meaning no output.

(which must again be DEVS models) and coupling relations [142, p. 150]:

$$N = (X, Y, D, \{M_d\}_{d \in D}, \{I_d\}_{d \in D \cup \{N\}}, \{Z_{i,d}\}_{i \in I_d}, \text{Select}) \quad (3.3)$$

where

X is a set of input events,

Y is a set of output events,

D is a set of component references (i.e. index set),

for each $d \in D$

M_d is a Classic DEVS model (i.e. sub-component of the coupling),

for each $d \in D \cup N$

I_d is the *influencer set* of d : $I_d \subseteq D \cup \{N\}, d \notin I_d$,

i.e. the set of components that influence the component d , and for each $i \in I_d$, $Z_{i,d}$ is the *i-to-d output translation function* specifying the coupling relationships:

$$Z_{i,d} : \begin{cases} X \rightarrow X_d, & \text{if } i = N \\ Y_i \rightarrow Y, & \text{if } d = N \\ Y_i \rightarrow X_d, & \text{if } d \neq N \text{ and } i \neq N \end{cases}$$

Finally, *Select* is the *tie-breaking function*,

$$\text{Select} : 2^D \rightarrow D,$$

that arbitrates the occurrence of simultaneous events by resolving which component is executed first.

It is to note that the coupling does not allow feedback loops where an output port of a component is connected to an input port of the same component, i.e. $d \notin I_d$.

For simulating DEVS models, a *simulator* (i.e. the simulation engine) has to implement steps to call atomic models, propagate output events and advance the simulation time. We will not go into specifics of DEVS simulators, as it is beyond the scope of this work. It should however be noted that there are several possibilities, one of the more intuitive approaches where the simulation engine directly maps the hierarchical structure of the model is described in [142] and [69]. This methods distinguishes between *simulator* components for executing atomic events and *coordinators* for routing event messages and invoking simulators inside couplings. Another approach described in [66] uses a flat simulation structure, thus avoiding additional message traffic between hierarchical layers of the simulation.

Parallel DEVS

A Classic DEVS model without *Select* function would not be uniquely defined when both an internal and external event occur at the same time. The *Select* function resolves this problem by allowing only one component to be activated at any time, thus avoiding such collisions [142, p. 140]. This resolution is performed however at the *coupling level*. In order to avoid the restriction of strictly sequential execution, a modification of Classic DEVS was proposed in [20] that handles collisions at the *atomic level*. The so-called *Parallel DEVS* (PDEVS) formalism replaces the *Select* function with a *confluent transistion function* δ_{conf} , leading to the following description of an *atomic* PDEVS [142, p. 143]:

$$PDEVS = (X, Y, S, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda, ta) \quad (3.4)$$

where

- X is the set of input event values,
- Y is the set of output event values,
- S is the set of state values,
- $\delta_{ext} : Q \times X^b \rightarrow S$ is the external state transition function,
- $\delta_{int} : S \rightarrow S$ is the internal state transition function,
- $\delta_{conf} : S \times X^b \rightarrow S$ is the confluent transition function,
- $\lambda : S \rightarrow Y^b$ is the output function,
- $ta : S \rightarrow \mathbb{R}_0^+ \cup \infty$ is the time advance function,
- $Q = \{(s, e) | s \in S, e \in [0, ta(s)]\}$ is the set of total states.

The confluent transition function δ_{conf} is executed when an PDEVS atomic receives external events at the same time of its internal transition, thus giving the modeller complete control over the collision behaviour. So, instead of serializing model behaviour (through a *Select* function), PDEVS leaves the decision of what serialization to use, if any, to the individual atomic. Zeigler [142] also suggests a default definition for the confluent transition:

$$\delta_{conf}(s, x) = \delta_{ext}(\delta_{int}(s), 0, x), \quad (3.5)$$

i.e. concurrent events are resolved by first carrying out the internal event and afterwards the external event. The *Extended DEVS* formalism [137], [138], a predecessor of PDEVS, even went so far and predefined this behaviour for its component models [20]. In contrast, PDEVS also allows other definitions to express special circumstances for concurrent events.

Avoiding the serialization imposed by the coupling presents a modeling advantage for the user in term of improved modularity and thus reusability as the behaviour of an atomic DEVS does not depend on its coupling environment (i.e. *Select* function). In addition, this improved independence also allows for the PDEVS atoms to be executed in parallel.

This parallelization however makes it necessary to introduce *bags*² of input events, denoted by X^b , to the external transition function. These bags can collect multiple input events (that may arrive at a certain point in time), thus recognizing that inputs can arrive in any order.

For *coupled* PDEVS models, the structure is almost identical to that for Classic DEVS, except that the *Select* function is no longer needed:

$$N = (X, Y, D, \{M_d\}_{d \in D}, \{I_d\}_{d \in D \cup \{N\}}, \{Z_{i,d}\}_{i \in I_d}) \quad (3.6)$$

with X , Y , D , I_d and $Z_{i,d}$ having the same meaning as for Classic DEVS coupled models (see page 34) and M_d is a *Parallel DEVS* model for each $d \in D$.

Hybrid PDEVS

So far, the DEVS and PDEVS formalisms can be used to describe discrete-event models. There also have been efforts to develop similar formalisms for continuous systems. The *Differential Equation System Specification* (DESS) [142] defines atomic as well as coupled DESS with purely continuous behaviour using differential equations. DESS is not described in detail here for reasons of brevity, for more details we refer to [102], [142].

For hybrid models including continuous as well as discrete behaviour, a hybrid classic DEVS formalism was proposed for the first time by Prähofer [100], [101] in the 1990s, which culminated later in the *Discrete Event and Differential Equation System Specification* (DEV&DESS) [142]. DEV&DESS combines the discrete specification of DEVS and the continuous specification of DESS.

However, as DEV&DESS builds upon Classic DEVS, it also inherits its shortcomings including the *Select* function. For this reason Deatcu et al. [27] proposed an adaption of Prähofer's formal hybrid DEVS definition to the PDEVS formalism. The specification, which we will denote with *hyPDEVS* in the following, includes discrete (X_d , Y_d) and continuous (X_c , Y_c) input/output values as well as states (S_c , S_d) and defines a hybrid atomic PDEVS as follows:

$$hyPDEVS = (X, Y, S, f, c_{se}, \lambda_c, \delta_{state}, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda_d, ta) \quad (3.7)$$

²Compared to a set of elements, a *bag* also allows multiple occurrences of an element, e.g. $\{a, b, c, a, b\}$ is a valid bag.

where

X is the set of inputs, consisting of X_c and X_d ,
 Y is the set of outputs, consisting of Y_c and Y_d ,
 S is the set of states, consisting of S_c and S_d ,
 $f : Q \times X_c \rightarrow S_c$ is the rate of change function,
 $\lambda_c : S_c \times X_c \rightarrow Y_c$ is the continuous output function,
 $c_{se} : S_c \rightarrow S_c$ is the state event condition function,
 $\delta_{state} : Q \times X_c \rightarrow S$ is the state event transition function,
 $\delta_{ext} : Q \times X_d^b \rightarrow S$ is the external state transition function,
 $\delta_{int} : S \rightarrow S$ is the internal state transition function,
 $\delta_{conf} : S \times X_d^b \rightarrow S$ is the confluent transition function,
 $\lambda_d : S \rightarrow Y_d^b$ is the discrete output function,
 $ta : S \rightarrow \mathbb{R}_0^+ \cup \infty$ is the time advance function,
 $Q = \{(s, e) | s \in S, e \in [0, ta(s)]\}$ is the set of total states.

The rate of change function f defines the continuous dynamics in terms of ordinary differential equations (ODEs). In addition to internal (i.e. time-driven) and external events (from discrete inputs X_d), state events (threshold events from continuous states) can be defined [98]. For this, the function c_{se} defines the conditions (in terms of zero-crossing) under which a state event is triggered. The state event is then carried out by computing the state event transition function δ_{state} . Since all three types of events may occur concurrently, the confluent transition function δ_{conf} needs to be adapted accordingly [27].

The specification for coupled models can be adapted from PDEVS (see equation (3.6) and [105]) with slight modifications. The input and output sets X and Y , respectively, now contain discrete as well as continuous values as well as the coupling relations $Z_{i,d}$ have to distinguish between discrete and continuous connections. This leads to the following specification for coupled hyPDEVS models:

$$N = (X_d \times X_c, Y_d \times Y_c, D, \{M_d\}_{d \in D}, \{I_d\}_{d \in D \cup \{N\}}, \{Z_{i,d}\}_{i \in I_d}) \quad (3.8)$$

The DEV&DESS formalism gives a similar specification for coupled DEV&DESS models, see [102], [142], except for the *Select* function.

One notable characteristics of hyPDEVS atomics is that their output function may in principle depend on the continuous inputs X_c , see also equation (3.7). In fact, two types of atomics are distinguished, in resemblance to state automata: While *Mealy* atomics define $\lambda_c : S_c \times X_c \rightarrow Y_c$, *Moore* atomics define a simpler output function $\lambda_c : S_c \rightarrow Y_c$ that does only depend on the state values S_c . Why that is relevant is because Mealy type atomics may cause algebraic loops in feedback couplings, i.e. a circular dependency of

output from input values, which constitutes an illegitimate model. Usually, the modeller has to prevent this scenario by taking care that each feedback coupling contains at least one atomic of Moore type. This legitimacy restriction is closely related to the legitimacy of atomic DEVS models, see equation (3.2).

These hybrid formalisms allow to specify continuous behaviour in term of differential equations alongside discrete behaviour (including state events). However hyPDEVS as well as DEV&DESS do not specify how to compute these differential equations during simulations. Different discretizations can be employed including common ODE solvers. In conjunction with DEVS, another discretization approach is quite popular, called *Quantized State System* (QSS), where not the time is discretized, but the state values. In particular, QSS calculates the point in time where a continuous state variable has changed more than a certain quantum. QSS is closely connected to DEVS as a continuous model discretized using QSS becomes essentially a discrete-event model, which can be mapped to an atomic PDEVS model [27]. In addition, the state-based quantization natively recognizes state events, which makes their handling easier [102]. For more details on QSS, we refer to [18], [67], [68]

An abstract simulation engine for hyPDEVS models has to incorporate handling discrete events as well as numerical integration algorithms and above all also the interactions between continuous and discrete parts of the model. Apart from the QSS approach, other simulator concepts exist, for example an ODE wrapper concept described in [27]. A prototype implementation in MATLAB, the so-called MatlabDEVS Toolbox, also provides a direct implementation of the hyPDEVS formalism [26], [97]. More details are given in section 3.1.2.

3.1.2 Tools for DEVS-based Hybrid Modeling and Simulation

There are a number of software tools for simulating DEVS models [35] which can probably be attributed to the popularity of DEVS in academia. Some of the more popular DEVS simulation tools include DEVJAVA [112], CD++ [136], DEVSim++ [124], DEVS-C++ [19], JDEVS [33] and ADEVS [93].

Most of these tools, however, build on Classic DEVS or PDEVS and focus on purely discrete models. In the context of this work, we are more interested in software tools that implement *hybrid* DEVS-based formalisms. Two of the more prominent examples are presented in the following.

PowerDEVS

PowerDEVS³ is an open-source software tool for DEVS-based hybrid systems modeling and simulation developed by Kofman et al. [69]. It builds upon the Classic DEVS formalism by embedding continuous behaviour using the QSS method (see also section 3.1.1).

³PowerDEVS is available at <https://sourceforge.net/projects/powerdevs/>

PowerDEVS aims at providing a DEVS simulation environment with a block-oriented graphical interface (that is similar to other established simulation tools) and library handling, also allowing use by non-DEVS-expert by hiding atomic DEVS definitions. For this, PowerDEVS is split into several independent programs, including a *model editor* containing the graphical user interface and an *atomic editor* for editing atomic DEVS models [11].

PowerDEVS implements some modifications of the Classic DEVS abstract simulator algorithm [142], including real-time simulation capabilities and a simplified specification for coupled DEVS models. For more details on PowerDEVS, we refer to [11], [69]

MatlabDEVS Toolbox

The MatlabDEVS Toolbox⁴ is developed by Deatcu et al. [97] and implements a PDEVS simulator with ports in MATLAB. It also offers experimental features for hybrid simulation, accompanied by the hyPDEVS formalism, which was presented in section 3.1.1. The MatlabDEVS Toolbox aims at bringing DEVS to the engineering community, where this approach is relatively unknown [97].

In fact, the MatlabDEVS Toolbox also implements an extension to hyPDEVS for variable structure systems [26], which is based on the *Dynamic Structure DEVS* (DSDEVS) specification [9]. As this extension is not relevant for our case study, we will leave this aspect aside and focus on the hybrid PDEVS simulator implementation.

The user requires a general understanding of the hyPDEVS formalism in order to be able to implement DEVS-based hybrid simulation models. Models are created in an object-oriented manner by implementing classes of hyPDEVS atomics, which can then be instantiated to create coupled models. In addition to coupling relations, also a *root coordinator* model has to be specified.

For computing the differential equations, the MatlabDEVS Toolbox provides on the one hand a mapping of a QSS algorithm into a PDEVS atomic. As an alternative, it also implements a novel ODE wrapper concept [27], where a closed representation of the continuous differential and algebraic equations is derived, in order to be computed with common MATLAB ODE solvers. This allows using a large number of established numerical methods for solving differential equations that are common in engineering applications, like implicit integration algorithms, predictor-corrector methods or adaptive step size control (for example, MATLAB's Runge-Kutta method `ode45` among others, see [126] for more details). The modular hierarchical model itself is not modified, therefore keeping the structural information intact.

During continuous simulation of an hyPDEVS model (cf. section 3.1.1), the ODE wrapper function (which is called by the ODE solver) calls the continuous output function λ_c , the rate of change functions f and the state event condition functions c_{se} of all hybrid

⁴The MatlabDEVS Toolbox is available at https://www.mb.hs-wismar.de/cea/DEVS_Tbx/MatlabDEVS_Tbx.html.

atomic subcomponents to calculate continuous outputs and derivatives and check for state events.

To allow interaction between discrete and continuous simulation, the *root coordinator* coordinating the simulation of the overall model was modified to operate in three phases: (i) initialization, (ii) discrete phase, and (iii) continuous phase. More details are given in [27]. The runtime execution is then as follows: Based on the minimum time stamp for the next internal event in any component, the root coordinator determines whether a discrete or continuous simulation phase has to be entered. In particular, if there is no imminent event, a continuous cycle is initiated that computes the differential equations until the time of the next event. At that time, the discrete phase is entered that computes all relevant atomic components according to standard (discrete) PDEVS behaviour. If all internal and external transitions have been executed for that particular time step, the simulation can progress by again entering continuous simulation first.

This way of handling the coordination between continuous and discrete simulation presents an important advantage, namely that the continuous simulator always knows beforehand when the next event occurs anywhere in the model⁵, in order to stop accordingly. This simplifies the underlying coupling scheme. If ODE computation is localized to an individual atomic component, it would have to be able to react to unforeseen external events (for example using rollback mechanisms), since it would then not be able to know beforehand when such an event occurs.

Implementing DEVS-based models using the MatlabDEVS toolbox also has some restrictions, especially regarding Mealy type atomics (see section 3.1.1). As Mealy type components need input values to compute output events, the user has to take care that these are provided accordingly, either by using Moore type atomics in between Mealy type atomics or by carefully considering the order of execution. Some further problems with hybrid DEVS formalisms regarding Mealy type blocks are also described in [102].

3.2 Design

3.2.1 Modular Hybrid Modeling Approach

In the course of the research project BaMa (Balanced Manufacturing, see [49], [70], [77], [80], a modularization approach was developed that aims at managing the complexity presented during system analysis of industrial production facilities. This approach divides the overall system from an energetic point of view into well-defined manageable modules, which then allow a focused system analysis independent from the surrounding environment. In the context of BaMa, these modular components are called *cubes* [70].

⁵This includes external as well as internal events, since every external event in DEVS is triggered by an internal event of another atomic. State events are not a problem in this context as they are detected by the ODE solver itself.

Figure 3.3 depicts an example configuration of cubes dividing a production facility. A cube can be, for example, a machine tool, a baking oven, a chiller providing cooling energy or the thermal zone surrounding the production.

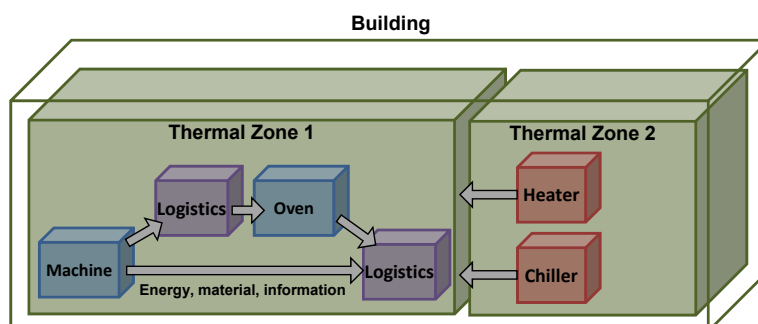


Figure 3.3: Example configuration of a production facility consisting of different cubes.

A cube represents a certain physical component with defined behaviour that interacts with its surroundings by exchanging energy, material and information. The cube concept defines these interfaces on an abstract level in order to ensure its applicability in a variety of production facilities as well as engineering domains. In particular, four different domains are distinguished: Machines and production processes, logistics, technical building services and building, see also figure 3.4. For more details, we refer to [70]

In a way, this extends the idea of an interdisciplinary reference model, which was applied in co-simulation case study (see section 2.2.1 on page 14) by unifying the approach of decomposition and the interface descriptions.

The cube approach also corresponds intuitively with the paradigm of component-based modeling, which allows to build larger models from well-defined components, meaning that cubes can also act as the building blocks of a simulation model. Here, this approach of modularization and decomposition promises reusability of model components [6], [7], which deems necessary in order to reduce the effort for developing and implementing simulation models in medium- to large-scale applications. Especially for interdisciplinary hybrid simulation models, this advantage cannot be underestimated.

3.2.2 Simulation Approach

The cube method described in the last section does not specify the way how to model the inner behaviour of cubes. One intuitive approach to model energy-related aspects is to draw energy balances around the borders of a cube and derive balance equations, leading to *continuous* model dynamics in terms of differential and algebraic equations.

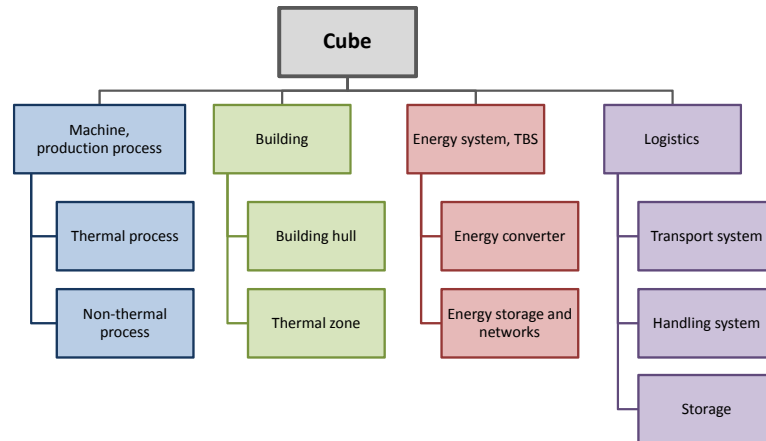


Figure 3.4: Different categories of cubes, divided into four areas: Machine and production process, building, energy system and technical building services, and logistics.

This description allows incorporating transient dynamic behaviour, which is essential when analyzing time-dependent interaction between different cubes.

On the other hand, it is necessary to model material flow using entity-based structures in combination with discrete events and states in order to be able to simulate persistent and traceable products (e.g. work pieces). This *discrete* behaviour can best be described using state machines.

These discrete and continuous aspects of a cube are often tightly coupled and interfere with each other, e.g. the temperature inside an oven may reach a certain point during heating, after which entities can be processed. Figure 3.5 illustrates this hybrid nature of Cube models, encapsulating discrete and continuous behaviour within their boundaries. The figure also shows again the three types of cube interfaces: material, energy and information.

Still, uniting these modeling techniques in the form of cube models presents one of the main challenges for implementation, which motivates the need for hybrid modeling and sophisticated formal descriptions as well as software solutions for component-based simulation.

The hybrid nature of cube models does not allow to easily split the overall hybrid model into discrete and continuous subsystems, that would enable using a co-simulation approach similar to the one used in our first case study (see chapter 2). Instead, we follow an approach for integrated hybrid simulation, that also promotes flexibility and reusability of models on a component level.

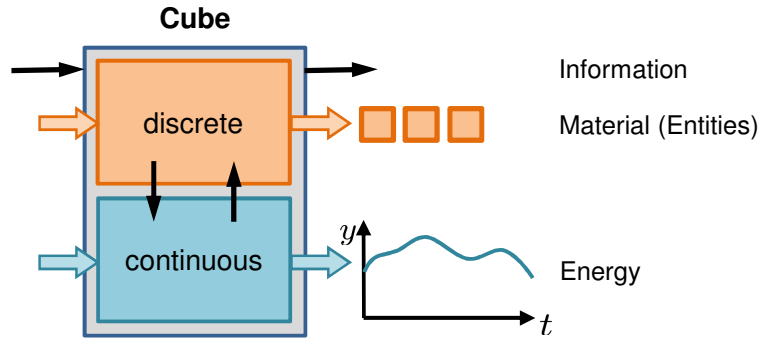


Figure 3.5: Hybrid nature of cube models encapsulating discrete material flow and continuous energy behaviour.

A hybrid DEVS formalism presents a suitable choice for modeling these cubes, because it also follows a component-based paradigm and allows for an open, transparent implementation.

First tests and an evaluation of PowerDEVS (see also section 3.1.2) proved unsatisfactory, mainly regarding simultaneous events, see [102], [103] for more details. The reason is that PowerDEVS employs a Classic DEVS simulation engine, which led us to switching to the hyPDEVS specification (which was presented in section 3.1.1).

3.3 Implementation

The implementation of a production facility case study was carried out using the MatlabDEVS Toolbox as simulation environment, see also section 3.1.2. As mentioned, the MatlabDEVS Toolbox implements the hyPDEVS specification based on Parallel DEVS. For solving the differential equations, we employ the ODE wrapper approach provided by the Toolbox.

3.3.1 Hybrid Model Component: Oven

As an example cube, we present the implementation of a simple conveyor oven model. We consider this a representative example for demonstrating the implementation as the oven model is truly hybrid, i.e. incorporating continuous as well as discrete aspects, and contains all important aspects of a cube model, i.e. differential equations, discrete entities, time-driven (internal) as well as external and state events, and a non-trivial state machine. Developing the model itself was not part of this work, only the DEVS-based implementation, which is why we will not go into detail regarding deriving the equations and state machine.

Model Description

The outer structure of the cube is composed by its interfaces over which it communicates with other cubes, which are shown in figure 3.6, see also [105].

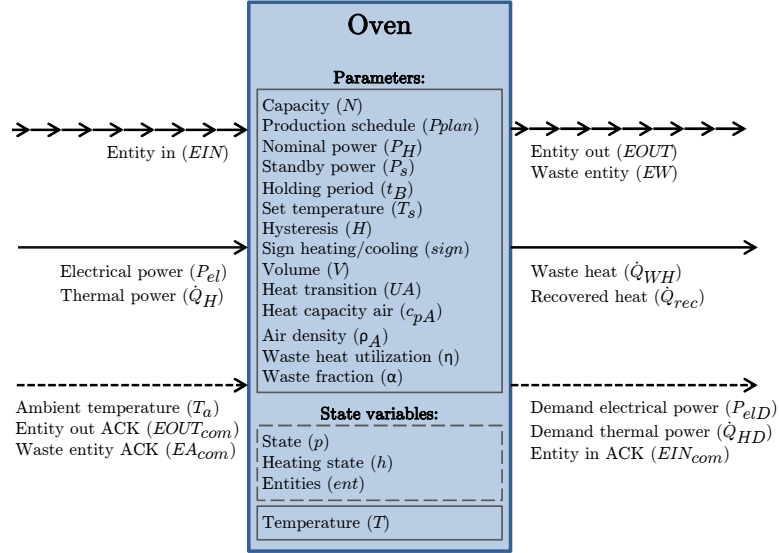


Figure 3.6: Interface of the presented oven model showing inputs (left) and outputs (right). The figure also shows parameters and state variables of the internal model.

For the internal model, the *continuous behaviour* is modelled by balance equations for energy-related variables, in particular the interior temperature T , waste heat \dot{Q}_{WH} and recoverable heat output \dot{Q}_{rec} :

$$\frac{dT}{dt} = \frac{\dot{Q}_H - (T - T_a) \cdot UA}{c_{pA} \cdot \rho_A \cdot V + \sum_{E \in ent} E.c_p \cdot E.m}, \quad (3.9)$$

$$\dot{Q}_{WH} = ((T - T_a) \cdot UA + P_{el}) \cdot (1 - \eta), \quad (3.10)$$

$$\dot{Q}_{rec} = ((T - T_a) \cdot UA + P_{el}) \cdot \eta. \quad (3.11)$$

where T_a denotes the ambient temperature, \dot{Q}_H the heating power input, c_{pA} the specific heat capacity, ρ_A the density of the thermal volume V , P_{el} the electric power input, η the heat recovery factor and UA denotes the heat transition through the oven walls. The term $\sum_{E \in ent} E.c_p \cdot E.m$ gives the sum of all heat capacities of all entities $E \in ent$ inside the oven.

The *discrete behaviour* of the oven governs the material flow, internal states and events and can be described semi-formally using state diagrams, see figure 3.7.

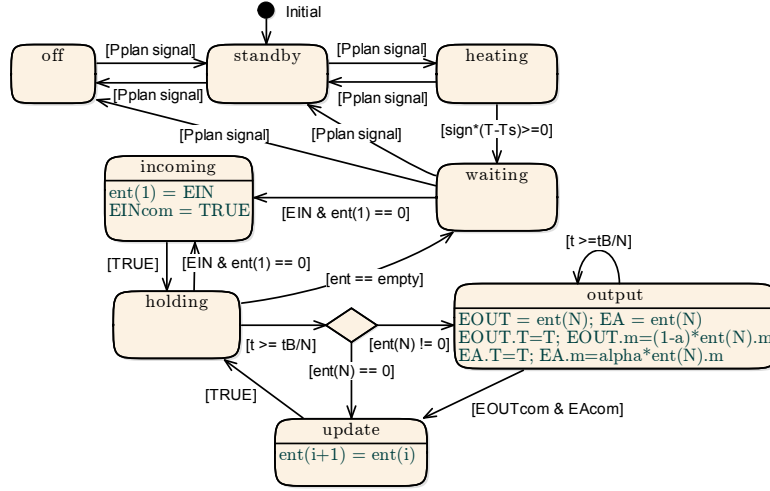


Figure 3.7: State diagram describing the discrete behaviour of the oven cube.

States are provided for off as well as standby mode, determined via `Pplan signal`. This `Pplan signal` comes from another component that reads the production schedule, which is provided by the user (input parameter, see also section 3.3.2). Similarly, the oven can be activated via a `Pplan signal`, after which it enters the state `heating`, where it requests energy for heating. When the requested temperature T_s is reached, state `waiting` is entered where the oven holds temperature and waits for incoming Entities (EIN). When an entity enters, the oven sends an acknowledgement signal (EINcom) and enters state `holding`, where processing takes place. In the meantime, further entities may enter (via state `incoming`). After the respective processing time (t_B/N) of the first entity has elapsed, the oven tries to send it on its output port (state `output`) and waits for acknowledgement (EOUTcom) from the downstream station. If this acknowledgement does not arrive in a timely fashion, the oven tries resending the same entity, and after receiving EOUTcom, it again enters state `holding` or `waiting`, depending on whether the entities are remaining or not. From the state `waiting`, the oven can be turned off, again via a `Pplan signal`.

Depending on a waste parameter α , the oven can also output a waste entity (output port EA) by splitting an outgoing entity into two, with the mass fractions $\alpha \cot E.m$ and $(1 - \alpha) \cot E.m$, respectively.

It is interesting to note, that, when looking again at the state diagram in figure 3.7, all three types of event transitions are present that are allowed when using hybrid DEVS-based formalisms:

- Time-driven (internal) event: Processing finished ($t \geq t_B/N$),
- External event: Incoming entity (EIN), among others,

- State-based event: Reaching internal temperature ($\text{sign} \cdot (T - T_s) \geq 0$)

The model equations also contain a user-defined parameter sign that allows to invert the thermal behaviour of the oven to essentially be operated as a cooler or freezer.

Translation into hyPDEVS

This semi-formal model description for the oven cube, which is independent from any DEVS-based implementation, now has to be translated into a hyPDEVS compliant model. As presented in section 3.1.1 and in particular equation (3.7), the hyPDEVS formalism defines an atomic components as the tuple

$$\text{hyPDEVS} = (X, Y, S, f, c_{se}, \lambda_c, \delta_{state}, \delta_{ext}, \delta_{int}, \delta_{conf}, \lambda_d, ta) \quad (3.12)$$

So as part of the modeling and translation process, one has to provide definitions for the respective functions specified by hyPDEVS, i.e. external transition function δ_{ext} , internal transition function δ_{int} , confluent transition function δ_{conf} , discrete and continuous output functions λ and λ_c , respectively, rate of change function f , state event functions c_{se} and δ_{state} and time advance function ta .

Unfortunately, this is not a trivial process, as several important model-related aspects have to be taken into account, which apply not only to the oven model, but all cubes [105].

Entity push semantics Consider for example the situation depicted in figure 3.8. If one wants to model exchanging entities between two stations A and B , the hyPDEVS specification only takes into account rudimentary event-driven behaviour, i.e. station A sends an entity in form of an event and station B has to process this event. However, if for example station B is not able to accept the entity because its storage capacity is reached, then it has to reject an incoming entity event. And in order for the entity to not be dropped and lost, the station B has to notify station A about the rejection so that station A can keep the entity. In terms of model robustness, it is preferable that instead of an rejection, each acceptance is notified to station A via an *acknowledgement* signal. As an alternative to this so-called *push semantics*, it would also be possible to employ *pull semantics* where station B always has to *request* an entity first before station A sends it. Nevertheless, there has to be some form of additional communication between the two stations, which is handles via a control information path between sender and recipient. In the tradition of common material flow simulation tools (e.g. Plant Simulation) and due to some severe issues⁶ with the pull principle, we decided to employ the push principle as the primary semantics for entity flow, although there are some exceptions (e.g. fetching entities from a storage). In addition to the recipient having to acknowledge each incoming entity, the sending station has to try to resend a rejected entity periodically in order to avoid a deadlock.

⁶If used excessively, the pull principle can reverse the control flow through the entire model. More details on this are also given in [105].

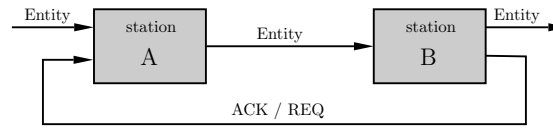


Figure 3.8: Simple example of two stations exchanging an entity, either using an acknowledgement signal (ACK) or request signal (REQ).

Input Buffer The described communication for exchanging entities also causes an additional iteration at constant time steps, during which messages are exchanged. Since a typical hyDEVS simulation engine will reset all external messages in between such iterations (since it assumes they have already been processed), even if they happen at the same simulated time step, it is a significant improvement for concrete applications to *buffer* incoming messages during iterations, at least as long as the time step does not change. This ensures that messages do not get lost during concurrent signals. The most prominent example of such a scenario also happens during entity exchange. Consider again the example in figure 3.8 and assume that at a certain time step station *A* has finished processing an entity, call it *x*, and tries to send it to station *B*. At the same time, station *A* already receives another entity, call it *y*, at its input for future processing. At this point, station *A* is not able to accept the entity *y* as it is still blocked from the entity *x* and has to wait for acknowledgement from station *B* before it can discard *x*. As long as the acknowledgement signal from station *B* comes in the same time step, station *A* might still be able to accept entity *y*. But for this, *y* has to be buffered during the entire exchange of entity *x*, otherwise it would be deleted from the input, because it has not been processed immediately.

Implicit Prioritization As mentioned in the beginning of the chapter (see section 3.1), Parallel DEVS (on which hyPDEVS is based) eliminates the *Select* function and with it the need to provide an explicit prioritization for executing atomic models, which opened the door for parallel processing. Unfortunately this also brings about modeling difficulties as the modeller can not make assumptions anymore about input events that occur at a particular time step also arriving concurrently, i.e. are available during the same iteration of the external transition function δ_{ext} . This problem has also been described in [102]. Also, during execution of δ_{ext} , incoming events have to be processed sequentially, thus imposing an implicit prioritization of input, depending on which is processed first. In the worst case, this can lead to some unintended behaviour and the modeller has to be very careful when implementing δ_{ext} .

Mealy Behaviour As mentioned, Mealy type atomics may cause difficulties, because they directly need an input signal for computing an output. The MatlabDEVS Toolbox also poses some restrictions in this regard, which is why for some connections a workaround had to be employed where a *Moore* type atomic, in particular a simple signal gain with factor 1, is connected in between two *Mealy* type atomics. This is also shown in

section 3.3.2. As said, this was necessary due to some software-specific restrictions, however the underlying problem concerns the formalism itself and is not trivial to resolve. Another peculiarity of the software concerns the number of continuous states necessary for an atomic component, for which dummy states had to be used in some cases. This can be seen in the source code in the appendix.

Finally, after considering the modeling aspects presented above, the hyPDEVs specification for the oven cube atomic model could be implemented. The individual functions, i.e. δ_{int} , δ_{ext} are not presented here for reasons of clarity, instead we refer to the source code in appendix A.

It is however interesting to note that during implementation we experienced that the modeller always has to be aware of how the model is executed, i.e. which function is called in which order. This was especially true for the output function λ and the internal transition function δ_{int} as they are mostly executed in tandem, as explained in section 3.1.1. The user can exploit this aspect for modeling, in some cases it is even unavoidable. One notable exception to this λ - δ_{int} combination may occur during concurrent events when δ_{conf} is executed immediately after λ . This may lead to additional problems, which is why we propose to simplify δ_{conf} for our application by only using the default behaviour (see also equation (3.5))

$$\delta_{conf}(s, x) = \delta_{ext}(\delta_{int}(s), 0, x),$$

This ensures that also in the δ_{conf} case, δ_{int} is executed immediately after λ , thus respecting the λ - δ_{int} combination.

3.3.2 Overall Model

To demonstrate the feasibility of the modeling concept and the implemented cube components, a simple example of a production plant was devised, shown in figure 3.9. The example was inspired by an industrial bakery that produces baked goods in different variants, fresh as well as frozen, and features a processing line with production and logistics components, an energy supply system, thermal zones and a building hull. Discrete material flow is incorporated as well as continuous energy flows and information signals. The example model is intended to include various critical aspects necessary for modeling production facilities, including: interaction of continuous and discrete characteristics, complex flow of entities (splitting, merging, batching, different paths) and scheduling of different product types.

In this example, the building cube contains four thermal zones, each representing a distinct part of the facility: production hall, cold storage, plant room and office. The thermal zones all have independent conditioning and exchange heat with each other via specified heat transfer parameters.

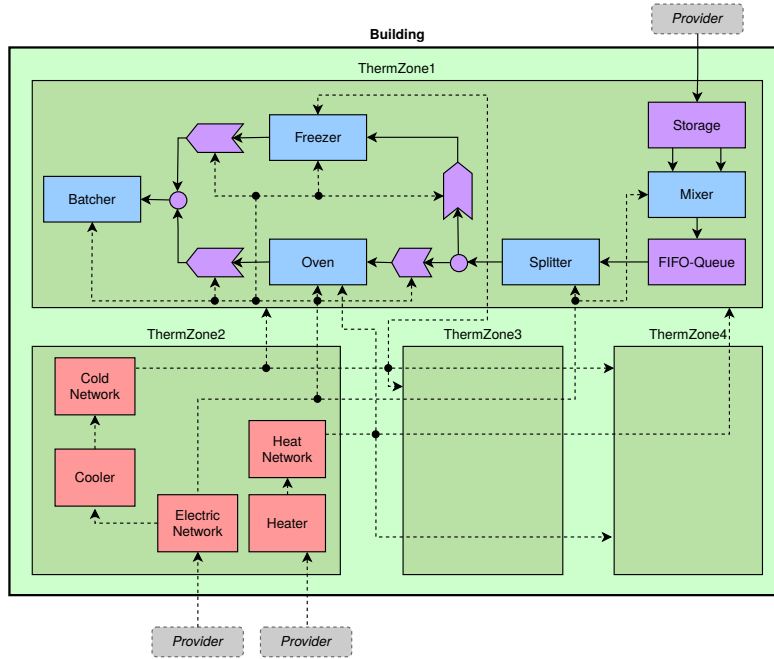


Figure 3.9: Example of a simple production facility, consisting of a processing line (top), energy system (bottom left) and thermal building cubes.

The energy system and energy network cubes are responsible for energy conversion and distribution of energy inside the system, in the form of heating, cooling and electric energy. This includes in particular a heater, chiller and energy grids with storage capacities.

For producing entities, respective ingredients are pulled from a storage and processed along the production line, including mixing, splitting and batching/packaging. The products either pass an oven for baking or a freezer for cooling, depending on the type of product.

For specifying simulation scenarios, the user defines production schedules, which are read as input parameters. Entries in the production schedule constitute commands for state changes (see for example the oven in section 3.3.1 and especially figure 3.7).

In addition to the production schedule, process parameters can be adjusted for different products based on a process sheet, e.g. oven temperature set point. Also, the simulation reads external data for weather conditions and ambient temperature.

The described example was realized in the MatlabDEVS Toolbox using the already implemented Cube atomic components. As mentioned, the toolbox employs an ODE wrapper for computing the differential equations using MATLAB's `ode45` algorithm.

According to the coupled hyPEDEVs specification (see section 3.1.1), the continuous and discrete coupling relations Z_{i_d} have to be defined for all cube instances. Taken the oven cube for example, the discrete couplings look as follows:

```

Zid_model = {
    % [...]
    % oven
    'belt1_stat','EOUT', 'oven','EIN';...    % entity input from
        previous station
    'oven','EINcom', 'belt1_stat','EOUTcom';... % acknowledgement to
        previous station
    'oven','EOUT', 'belt3_stat','EIN';...    % entity ouput to next
        station
    'belt3_stat','EINcom', 'oven','EOUTcom';... % acknowledgement
        from next station
    'oven','EA', 'sink_waste','EIN';...      % waste entity output
    'sink_waste','EINcom', 'oven','EAcum';... % acknowledgement from
        waste output
    % [...]
}

```

Similarly, the continuous couplings for the oven are specified using the continuous output translation function:

```

CZid_model = {
    % [...]
    % oven
    'e_grid',8, 'gain23',1;...    % Pel input (via gain)
    'gain23',1, 'oven',1;...
    'heatinggrid',3, 'gain1',1;... % Q_H input (via gain)
    'gain1',1, 'oven',2;...
    'therm_zone1',1, 'oven',3;... % Ta input
    'oven',4, 'heatinggrid',4;... % Q_HD output
    'oven',3, 'e_grid',9;...      %PelD output
    'oven',1, 'therm_zone1',12;... %Q_WH output
    % [...]
}

```

Unlike in Zid, where the ports of the atomics could be accessed directly via name, here the ports have to be accessed via index number instead. This is due to an immaturity of the MatlabDEVS Toolbox in this regard and may as well be fixed in future releases.

As mentioned, some of the continuous connections need to use gains in between in order to resolve Mealy type behaviour.

3.3.3 Validation

Individual aspects of the overall model have been validated using independent implementations, which were developed as part of the research project BaMa. The continuous sub-model including thermal zone cubes and energy systems components were implemented and tested using Dymola and show satisfying consistency. The hyPDEVS-based implementation of the oven cube could also be validated against an independent implementation in Dymola, which uses native data structures for representing entities. See

[104] for more details. For validating the flow of entities, entry and exit times were compared to independent calculations and also show satisfying agreement.

3.4 Testing and Results

In the following, exemplary simulation results are presented from different scenarios, in order to demonstrate the application of the implemented case study model. Parts of these results have also been presented in [105].

Production schedule and process sheet determine the production scenario and serve as input parameters for the simulation. The simulation also reads external input data to factor in ambient temperature conditions.

Table 3.1 shows the production schedules for two considered scenarios, which take place over one day (00:00 to 24:00). Scenario 1 and 2 are intended to provide a comparison of two different production configurations under the same conditions, i.e. the same number of entities have to be produced by the end of the day.

Table 3.1: Production schedules for two example scenarios

Station	Time	State	Product type	Quantity
Scenario 1				
Storage	02:30	prepare	1	8
	10:00	prepare	2	16
Production	03:00	on	1	
	10:30	on	2	
	24:00	off	-	
Oven	00:00	heating	1	
	10:00	off	-	
Freezer	00:00	cooling	2	
	24:00	off	-	
Scenario 2				
Storage	00:30	prepare	1	8
	06:00	prepare	2	16
Production	01:00	on	1	
	06:30	on	2	
	16:00	off	-	
Oven	00:00	heating	1	
	07:00	off	-	
Freezer	06:00	cooling	2	
	15:30	off	-	

3. DEVS-BASED MODELING CASE STUDY

As a result of the simulation, figure 3.10 presents a comparison of the entity flow between the two scenarios.

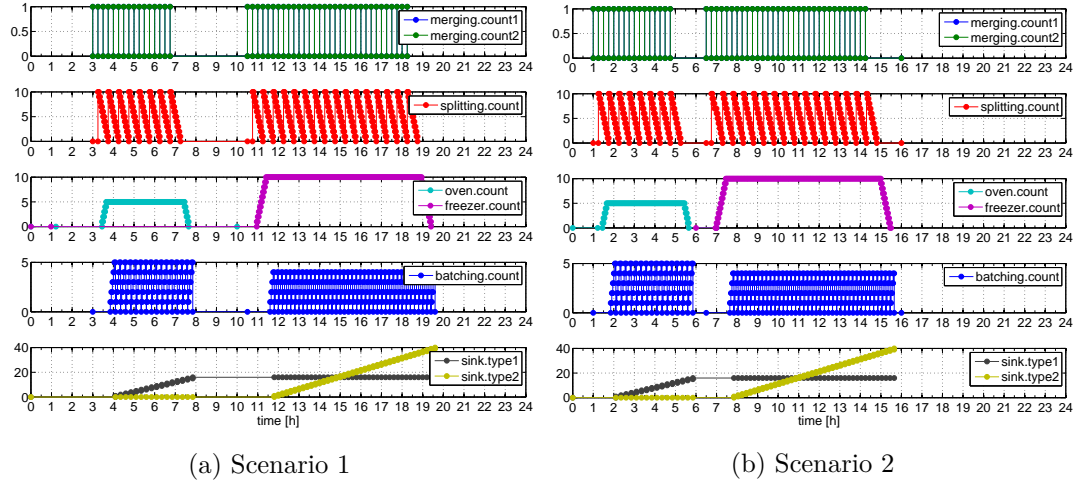


Figure 3.10: Comparison of entity flow: Number of entities over time in the different stations.

Figure 3.11 depicts the total energy demand over time for the two scenarios. The main difference in the electric energy demand can be attributed to the lower power consumption of the production line in scenario 2 in that it is switched off earlier after production has finished. Heating and cooling demand present only a marginal difference due to slight different operating times for oven and cooler (see also figure 3.12), but the main energy demand stems from conditioning the thermal zones, which is the same for both scenarios.

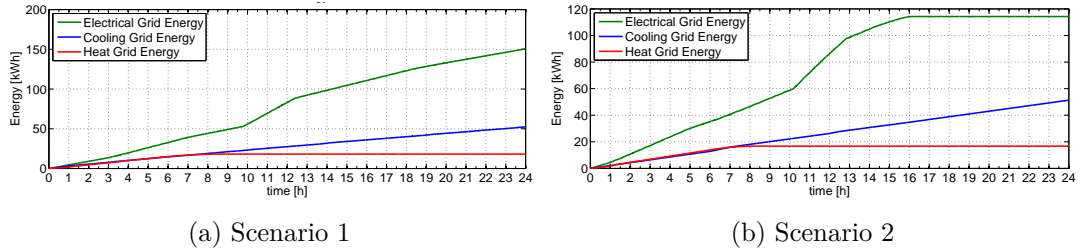
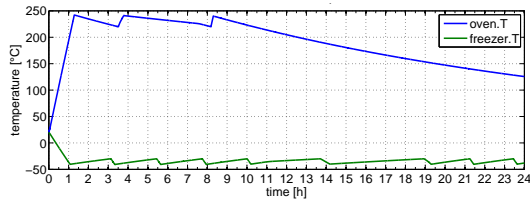


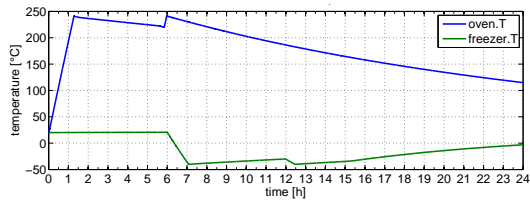
Figure 3.11: Comparison of energy demand: Energy consumption for heating, cooling and electric energy over time.

In the comparison of temperature profiles for oven and freezer in figure 3.12, one can see that, due to the production schedule being finished earlier in scenario 2, the stations can be turned off earlier, thus preserving energy. The temperature fluctuations during the day can be attributed to the two-point controller.

Finally, figure 3.13 depicts the profiles for thermal zone temperatures and ambient air temperature. There are no significant differences visible, as the thermal zone temperatures



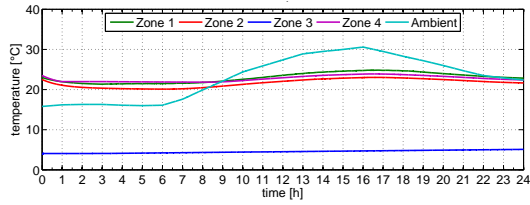
(a) Scenario 1



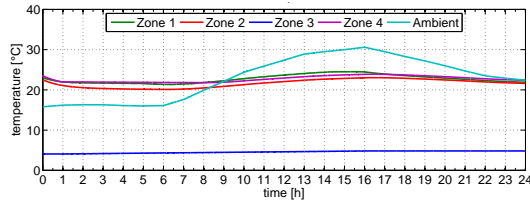
(b) Scenario 2

Figure 3.12: Comparison of oven and freezer operation: Temperature profile for oven and freezer over time.

mainly depend on the ambient temperature (which is the same for both scenarios) and are only marginally influenced by waste heat from production machines.



(a) Scenario 1



(b) Scenario 2

Figure 3.13: Comparison of thermal zone temperatures: Temperature profile for thermal zones as well as ambient temperature over time.

Evaluation and Comparison

After presenting both the co-simulation and DEVS-based modeling approach and their practical application in case studies, we now want to address their feasibility with regard to relevant criteria and compare them with each other. This evaluation is intended to help choosing which of the fundamental modeling approaches is more suitable for specific application use cases.

First, we present some related work regarding evaluation and selection of simulation software and a description of chosen evaluation criteria. Afterwards, both the co-simulation and DEVS-based modeling approach are evaluated and compared, followed by some concluding remarks.

4.1 Related Work

Often simulation tools are evaluated either on their own merits, or in comparison with other tools [90]. Not only is selecting a suitable simulation tool from a possible set of alternatives in the presence of evaluation criteria typically a difficult complex multi-criteria decision problem [4], [8], [60], but the large number of available tools adds an additional scale to the problem. For this reason, several studies exist in research literature concerning the selection of simulation software as well as evaluation techniques and criteria. These can provide assistance during the decision making process and help to manage a multitude of often intangible criteria.

Selection Methodologies

The authors of [125] describe a two-phase selection methodology for simulation software selection. The first phase checks a wide list of potential candidates for availability of the most important features and selects a subset based on specific criteria, while in the second phase the remaining candidates are analysed and evaluated in detail [4]. Various

methods can be used for evaluation in both phases, for example a method for extracting and categorizing evaluation criteria like described in [90].

In [117], a framework is presented for choosing discrete-event simulation software. Starting from a project objective, it addresses model use, dissemination and modeling range, among other aspects.

Evaluation Techniques

Banks [8] suggested some considerations to be made regarding various criteria for selecting simulation software and proposed a simple scoring model where values between zero and ten are assigned to each criterion, which are then summed and normalized. In [134], the authors compare four different manufacturing simulation tools by rating various groups of criteria on a scale from one to ten.

The *Analytic Hierarchy Process* (AHP) [43], [107] provides an evaluation technique [24] in form of a systematic approach for structuring and assessing complex multi-criteria decision making problems. The technique originally stems from socio-economic research and aims at reducing inconsistencies in human judgement. The assessment is based on a multilevel hierarchical decomposition and quantification of objectives, interrelated criteria and alternatives [119]. It provides a means to measure especially intangible factors by using pairwise comparison matrices that quantify the relative importance on an absolute scale of one element over another with respect to a common property. AHP is widely used for solving decision making problems in various fields [63], [73], [119].

Although less prominent in literature, the *Analytic Network Process* (ANP) is a generalisation of the Analytic Hierarchy Process for decision problems that cannot be structured hierarchically [43], [108], [109]. For example, Ayağ [4] proposed an ANP-based approach to evaluate a set of simulation software alternatives.

Evaluation Criteria

Hlupic et al. [56] provide a comprehensive framework for selecting evaluating simulation software, containing more than 310 criteria, which is intended to be applicable to any simulation package regardless of its application area. It presents an advancement from earlier research in the area of manufacturing simulators [55], [57], [59]. A comprehensive description of all criteria is given in [57]. This evaluation framework has been tested through several case studies. It is claimed that dozens of simulation specialists have already used this evaluation framework for software selection [15].

Also based on this work, Nikoukaran et al. [89], [90] classified criteria for simulation software selection into a hierarchical structure. However, they also do not aim to provide a weighting of importance between these criteria.

A simplified framework is presented in [15], including most important features and proposed guidelines to be used by non-experts. The features and guidelines are derived

from practical experience and survey of literature and focus in particular on Business Process Modeling (BPM).

In addition, a range of other authors proposed preferred lists of features and criteria with varying range, classified into groups, e.g. [8], [24], [31], [60], [74], [75], [81], [90], [118], see also [91] and [1] for an overview. In general, these lists overlap substantially, however, since these criteria are usually intangible, the interpretations of these criteria often vary.

Tools for Selecting Simulation Software

The software tools *SimSelect* [58], based on the works of Hlupic et al. [55]–[57] uses a database of evaluated software tools and user-specified priority needs to suggest suitable simulation software and possible alternatives.

Similarly, *Smart Sim Selector* [45] was developed for the purpose to provide user support for selecting simulation software, using three different ranking techniques, including AHP.

4.2 Scope of the Evaluation

Evaluation of simulation software is often carried out on the basis of case studies [55], [64]. The steps involved in a qualitative evaluation include formulating the research question, collecting relevant data and interpreting the data in order to draw conclusions regarding the research question [42], [64], [133].

Regarding this thesis, the research question was already formulated at the beginning of this work (see section 1.2). The collected data are based mostly on the presented case studies and the experience gathered during that time and what could be derived from literature.

We intentionally focus our evaluation on the modeling methods and related aspects instead of specific simulation tools, as both of the simulation tools employed in the case studies (BCVTB and the MatlabDEVS toolbox, respectively) are currently still in a prototype phase, which does not allow to make a fair and representative assessment of practical software-related aspects, such as (graphical) user interface, visualisation features, software costs or compatibility with other tools, like it is presented in most of the literature mentioned above [56].

We also do not intend to judge the importance of individual criteria with respect to one another as this typically depends on a concrete application scenario and is inevitably subjective [15]. Instead, we focus on a qualitative evaluation and comparison [42] of the modeling features, advantages and disadvantages co-simulation and the DEVS approach entail, specifically in the context of interdisciplinary simulation of production systems.

Instead of directly comparing the two modeling approaches, we first want to evaluate them individually on an absolute instead of relative scale. The evaluation is based on own experience gathered during the two cases studies, which are considered to be

representative for interdisciplinary modeling of production systems as they both contain all major aspects and areas of investigation, i.e. production machines, logistics, building as well as energy infrastructure.

In order to come to a final conclusion on which modeling approach to choose for a particular application, one may use the qualitative results presented in this work and additionally assign different levels of importance (like presented in [8]) or use for example a more fine-grained method for quantification like the Analytic Hierarchy Process (AHP) (see section 4.1).

4.3 Criteria

Based on a literature survey [8], [15], [24], [31], [56], [57], [60], [74], [75], [81], [90], [118], [128] as well as practical experience, table 4.1 provides a list of criteria, that serve as the basis for the following evaluation. The list has been adapted and reduced to reflect the scope of our evaluation and aims to be as objective as possible. It includes basic requirements, that are deemed necessary, especially in our focus of interdisciplinary simulation in an industrial context.

The criteria are grouped into three categories that try to reflect a typical three-tier architecture of software for modeling and simulation (see [111] for example).

The qualitative scale is divided into three steps (difficult/medium/easy, low/medium/high, etc.), which are sorted from worst to best. In addition, we introduce two intermediate steps for a more fine-grained assessment, resulting in the five-point scale shown in figure 4.1.

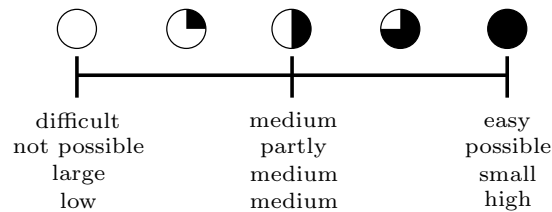


Figure 4.1: Five-point scale for assessment of evaluation criteria.

The following sections give a more detailed explanation of the individual criteria.

4.3.1 User Considerations

This category is concerned with evaluation criteria from a user perspective and includes:

- *Ease of use: Model development (expert)*: This criterion describes how easy it is for an expert user to develop and implement a new model using the modeling approach under consideration.

Table 4.1: Evaluation criteria including qualitative scales, grouped into user considerations, modeling capabilities, and simulation performance.

User considerations	Ease of use: Model development (expert)	Ease of use: Simulation user (non-expert)	Ease of learning	Ease of debugging	Modeling transparency	Collaboration
	difficult medium easy	difficult medium easy	difficult medium easy	difficult medium easy	low medium high	difficult medium easy
Modeling capabilities	General features	Time effort for model building	Hierarchical model building	Modularity	Model reusability	Flexibility
	none some all	large medium small	not possible partly possible	not possible partly possible	difficult medium easy	difficult medium easy
	Libraries with pre-defined components	Level of detail	Maintainability	Model import/export	Transferability	Separation of modeling and simulation
	not prov. user-def. provided	low medium high	difficult medium easy	difficult medium easy	difficult medium easy	not possible partly possible
Simulation performance	Performance efficiency	Accuracy	Reliability	Use of established algorithms	Data handling	Distributed simulation
	low medium high	low medium high	low medium high	not possible partly possible	not prov. partly provided	not possible partly possible

- *Ease of use: Simulation user (non-expert)*: Likewise, how easy is it for a non-expert user to use a finished simulation model, change parameters or execute simulation runs?
- *Ease of learning*: How easy is it for a novice user to learn the basic aspects to be able to employ the modeling approach for various tasks?
- *Ease of debugging*: During model development and use, how much effort is necessary for finding bugs and errors in the models?
- *Modeling transparency*: To which degree is an (expert) user able to gain insights into the modeling language, simulation algorithm, etc. in order to understand and trace the underlying simulation process.

- *Collaboration*: How easy is it for multiple users to work together on the same model, either simultaneously, sequentially, or otherwise? This aspect is especially relevant when considering interdisciplinary investigations, where experts from different fields need to work together towards a common goal.

4.3.2 Modeling Capabilities

The category *modeling capabilities* lists several criteria relevant for model development, implementing as well as maintaining simulation models.

- *General features*: This criterion asks if all modeling features are provided which are necessary for interdisciplinary modeling of production systems, including discrete entities, energy flows or hybrid dynamics.
- *Time effort for model building*: This criterion estimates the overall time needed to implement a model. This depends on many other factors, such as ease of use and the complexity of the system being modelled [57].
- *Hierarchical model building*: Is it possible to compose models hierarchically, thereby improving structure and clarity of the model?
- *Modularity*: Does the modeling method allow to encapsulate arbitrary parts of the model into separate modules? This not only promotes reusability, but also developing models step by step for easier debugging.
- *Model reusability*: This evaluates whether a modeling approach enables reuse of models created previously, which improves modeling efficiency, model quality and time needed for model building [15], [57].
- *Flexibility*: Is it possible for the modeller to add custom logic, e.g. as user-defined pieces of code? This enables to model a variety of different types of systems instead of being limited to a set of pre-defined components [56].
- *Libraries of pre-defined components*: Does the modeling approach allow making use of already implemented and ready-to-use components, for example libraries offered by the open-source community or third-party vendors? Such libraries are typically offered in conjunction with a particular software tool.
- *Level of detail*: Which level of detail can be incorporated into a model and how much effort does that take? This is also connected to the modeling flexibility criterion.
- *Maintainability*: This criterion asks the level of effort necessary to maintain, adapt and extend a model after its initial use by editing individual parts of the model.
- *Model import/export*: Is it possible to use the model or parts of the model in other areas or to import models previously created for a different purpose?

- *Transferability*: According to [128], transferability stands for the broad applicability of the general approach for different cases and purposes with reasonable effort, for example for different industries or to answer different research questions.
- *Separation of modeling and simulation*: Is it possible to separate the model implementation from the simulation algorithm, for example in order to be able to test different algorithms for the same model.

4.3.3 Simulation Performance







In the category *simulation performance*, we address issues regarding simulation execution and use.

- *Performance efficiency*: This criterion asks the ratio of simulation speed to model complexity. Questions shall be addressed such as how much overhead is involved due to the modeling approach that slows down model execution. An efficient simulation enables for example to employ simulation-based optimization techniques, which usually involve a large number of simulation runs.
- *Accuracy*: This criterion estimates the level of accuracy with which the simulation results can be provided in general. This also relates to numerical stability for example.
- *Reliability*: How often is a model that is implemented using a certain modeling approach prone to unexpected behaviour?
- *Use of established algorithms*: Is it possible to use established and trusted simulation algorithms (for example ODE solvers) for simulation?
- *Data handling*: How easy is it to provide facilities for data storage, retrieval and manipulation? This involves for example managing simulation results, but also parameter settings and input data. This criterion like the others aims not so much on the concrete simulation software as it does on the modeling approach that facilitates a certain kind of software architecture.
- *Distributed simulation*: Does the modeling approach allow for distributed simulation in general, in order to not only facilitate parallel execution, but also distributed model development?

4.4 Evaluation of Co-Simulation

In the following tables 4.2 to 4.4, we present our assessment of the evaluation criteria for the co-simulation modeling approach, based on practical experience gathered during the case study as well as from relevant literature. The criteria are assessed individually and independently from the evaluation of the DEVS-based approach in section 4.5 and is based on the five-point scale presented in figure 4.1.

Table 4.2: Evaluation of user consideration for the co-simulation modeling approach.

Criterion		Assessment
Ease of use: Model development (expert)		Model developers can, for the large part, resort to established simulation tools. However, all developers involved need to agree on a common interface structure and expert knowledge is required to establish the simulator coupling.
Ease of use: Simulation user (non-expert)		Usually, multiple tools have to be executed during a simulation run. Parameters for scenario-based simulation are often scattered across multiple sub-models.
Ease of learning		Established simulation tools are often easier to learn. On the other hand, middleware solutions for co-simulation are still experimental and require more background knowledge.
Ease of debugging		Experience showed that co-simulation, where several software tools run in parallel, are difficult to debug, not at least because possible error messages from clients tend to get lost if not adequately taken care of by the server. For debugging sub-models, one can usually rely on the given simulation environments.
Modeling transparency		It is usually difficult to trace the simulation process in detail, since it is distributed across several clients. In addition, commercial simulation clients often do not provide access to the underlying modeling language and the user has to believe that the sub-models work as intended.
Collaboration		Co-simulation of course enables multiple engineers to collaborate together on a co-simulation model. However, significant communication is necessary to agree on common interfaces, which is a non-trivial and error-prone task.

Even though co-simulation allows to use established simulation software in principal, experience showed that typically not all features this software offers can be exploited fully during a co-simulation, for example data visualisation and animation capabilities are limited to the particular sub-model. These aspects reduce the initial advantage of co-simulation from a user perspective.

Table 4.3: Evaluation of modeling capabilities for the co-simulation modeling approach.







Criterion		Assessment
General features	●	By combining adequate simulator clients, all necessary features can be offered in general, including product entities and state machines (discrete-event simulator) or energy flows (continuous simulator).
Time effort for model building	◐	Sub-models can be developed quickly using established functionality of respective simulation environments, especially if libraries of pre-defined model components are available. Establishing the coupling relations on the other hand can be cumbersome and error-prone, especially when using low-level data communication mechanisms (like BCVTB).
Hierarchical model building	◐	For the sub-models, individual simulators may allow hierarchical composition. However, keeping a clear hierarchy for the overall model is usually difficult due to inherent modeling restrictions. For example, when using one continuous and one discrete simulator as part of a co-simulation, all continuous and discrete parts of the overall model have to be divided into the respective simulator.
Modularity	◑	Similar to the hierarchical model building, achieving modularity for sub-model components is typically not a problem. However, a modular implementation for components spanning several simulators (because they contain continuous as well as discrete dynamics, for example) is usually not possible. One could perhaps divide the overall model into smaller sub-models, each with its own simulator, thereby improving modularity. This would, however, also increase communication overhead significantly.
Model reusability	◐	Related to the modularity, individual sub-models and respective components can in principle be reused. Reusing coupling relations on the other hand is typically not effective.
Flexibility	◑	Flexibility in terms of user-defined logic usually concerns the individual sub-models and the functionality of the respective simulation clients. The co-simulation coupling may however pose some restrictions due to its limited flexibility.
Libraries with pre-defined components	◑	Individual simulation environments may provide a substantial basis of ready-to-use model components (like for example Modelica/Dymola in our case study). These are, however, limited to the individual sub-models and typically do not involve coupling interfaces.

(continued)

Level of detail	●	Modeling capabilities of individual simulation tools are often tailored to specific application domains (e.g. electrical circuits, mechanical systems) and thereby provide sufficient level of modeling detail.
Maintainability	◐	Due to the distributed nature of the co-simulation model, it is difficult to maintain, extend and adapt the model for later use. This difficulty is increased by the fact that model alterations often involve adapting the coupling relations as well, which was cumbersome and error-prone in our case study.
Model import/export	◑	This presents one significant advantage of co-simulation, namely that sub-models can be imported from other areas. For example, in our case study the building sub-model was developed using Building Information Modeling (BIM) methodology and tools, and could later on be used for co-simulation as well. In the same way, sub-models may later be used for stand-alone simulations with little modifications. Mainly the coupling interfaces need to be altered during model import/export.
Transferability	●	Transferability is different from reusability since – even though models may not be reusable – the generic modeling approach is also applicable for various other domains of interest and mainly depends on available simulator clients and their functionality.
Separation of modeling and simulation	◐	This also depends heavily on the possibilities of the client software. Common simulation software typically allows separation of the simulation algorithm from the implemented model, in order to allow using different algorithms for the same model. However, the coupling mechanism for the co-simulation – which may be viewed as part of the simulation algorithm – is usually embedded into the model. Thereby it is not easy to employ a different coupling strategy without significant modifications.

One has to be careful in this regard to distinguish between modeling capabilities that stem from the particular approach and the features that are provided by the software. For example the High Level Architecture technology (see section 2.1.3) provides slightly improved reusability of simulation models according to [7]. But nevertheless, the basic approach of having to divide hybrid model components onto several simulators discourages reusability on the modeling level.

Table 4.4: Evaluation of simulation performance for the co-simulation modeling approach.







Criterion		Assessment
Performance efficiency		Although co-simulation allows using multiple simulation algorithms, each tailored to the characteristics of the individual sub-models, it also involves substantial communication overhead. In addition, the modeling methods and level of detail used for the sub-models are typically designed for stand-alone simulation, and thus may bring about additional model overhead if this level of details is unnecessarily high for co-simulation investigations. This was evidenced in our case study (see section 2.3.1) and required careful consideration regarding the level of modeling detail.
Accuracy		Numerical accuracy was not studied in detail as part of the case study. However, in chapter 2 it is mentioned that the discretization of communication intervals and necessary extrapolation of input signals lead to additional numerical errors, depending on the coupling strategy and length of the communication interval. Therefore, the communication step size (which is usually fixed) has to be carefully considered by the user.
Reliability		The interaction of multiples software tools may cause some unexpected behaviour at some point, especially if the communication is not implemented and executed in a transparent manner.
Use of established algorithms		As mentioned, the ability to employ algorithms of established simulation tools, constitutes one of the major advantages of co-simulation.
Data handling		Data handling in co-simulation is a non-trivial task as the data is typically scattered across multiple simulators, not all of which the middleware has direct access to. Either the simulator clients have to provide access to the remaining data or a superordinate software instance has to manage and merge the data from the sub-models.
Distributed simulation		The communication between the simulators of course enables distributed simulation of the sub-models, which in turn also facilitates user collaboration. Depending on the coupling strategy (Jacobi type of loose coupling for example), the sub-models may also be executed in parallel, thereby increasing simulation speed.

Due to the sensitivity of the simulation accuracy on the communication interval, there are studies that have derived guidelines on how to choose an appropriate communication step size, see [135] for example.

4.5 Evaluation of hybrid DEVS-based Modeling

Tables 4.5 to 4.7 show the results of the evaluation for the DEVS-based modeling approach. Like in the previous evaluation of the co-simulation approach, the criteria are assessed individually and independently and presented in the five-point evaluation scale.

Table 4.5: Evaluation of user consideration for the DEVS-based modeling approach.

Criterion		Assessment
Ease of use: Model development (expert)		Initial model development requires substantial expert knowledge due to the small number of features inherently provided by the hyPDEVS formalism. The user has to manually implement additional mechanisms to aid more specialized and high-level features, like for example entity push semantics or input buffers, as described in section 3.3.1.
Ease of use: Simulation user (non-expert)		Due to the integrated nature of the completed model after implementation, subsequent use by non-experts becomes significantly easier. The software is able to provide a uniform and consistent user interface for setting parameters and defining scenarios.
Ease of learning		The formal approach to modeling requires a substantial learning phase (compared to graphical model building for example) [57], especially for users with limited modeling experience in general. Although hyPDEVS itself is fairly intuitive, applying the formalism in a concrete example requires some experience and sometimes brings unintuitive results (for example due to the separation of output and internal transition function).
Ease of debugging		Integrated modeling of all relevant aspects also facilitates easier debugging in general. However, the sometimes unintuitive model implementations require a basic understanding of the DEVS simulation engine during model debugging.
Modeling transparency		Because the DEVS specification itself as well as typical simulation engines are open knowledge, it is easy to gain deep insight into the inner workings of the simulation.
Collaboration		Although it is possible to employ a modular approach where different modules are developed and implemented by different people (for example different domain experts), they all need a basic understanding of the modeling formalism. In general, the communication between the different experts has to be tighter than when using co-simulation.






Regarding the DEVS-related usability issues, one must not get blinded by the fact that

the MatlabDEVS Toolbox itself is still a prototype, because these issues for the most part concern the underlying formalism. They are a result of the generic and low-level nature of hyPDEVS, which does not pose many restrictions on what can be modelled on the one hand, but on the other hand also does not provide support for more specialized modeling features (e.g. robust handling of persistent entities, like described in section 3.3.1).

Evaluation of modeling capabilities for the DEVS-based modeling approach.

Criterion		Assessment
General features	●	The hybrid nature of the hyPDEVS formalism allows to model all necessary features, including state machines, entities, continuous dynamics for energy flows, etc.
Time effort for model building	◐	Building atomic models initially requires substantial time effort and expert knowledge, see also the criterion <i>Ease of use: Model development</i> . As all necessary features can be provided in general, many aspects have to be implemented by the model developer, since they are not inherently supported by the formalism.
Hierarchical model building	●	The hyPDEVS specification is designed to allow hierarchical model composition.
Modularity	●	Along the line of hierarchical model building, modularity is one of the features inherently supported by the DEVS specification. And hyPDEVS in particular also allows modular encapsulation of hybrid components that involve continuous as well as discrete dynamics.
Model reusability	●	Improved modularity and hierarchical model building also facilitate better model reusability, as hybrid components can be encapsulated on every level of granularity in order to make them accessible for reuse.
Flexibility	●	As difficult as the low-level approach of DEVS it makes for the model developer to implement atomic models, it also allows maximum flexibility in regard to customization. The simplicity of the hyPDEVS formalism makes very little restrictions on what is possible to model.
Libraries with pre-defined components	◐	Currently, there are almost no libraries available with ready-to-use DEVS-based components. However, the prominent modularity and model reusability allows model developers to create such libraries themselves for reuse, thereby potentially reducing the modeling effort for subsequent implementations.

(continued)

Level of detail		Although it is possible in general for the model developer to implement models with an arbitrary level of detail, it demands substantial effort, since no domain-specific modelling support is given by the simulation environment. On the other hand, the modular approach of hyPDEVS facilitates developing models iteratively with increasing level of detail, since modular components can be exchanged for more detailed ones, without affecting the rest of the model.
Maintainability		Due to the tight integration of all aspects on the modeling level, it becomes easier for model developers to maintain, adapt and extend their models.
Model import/export		The specific model description and simulation engine of the hyPDEVS formalism make it difficult to combine DEVS-based models with other model implementations. In theory, it would be possible to specify translation processes from other model description into hyPDEVS and vice versa, in order to make them accessible for model import/export. There are some studies in this regard, which are related to DEVS [22], [121], however few of them focus on hybrid continuous/discrete modeling and there are almost no readily available tools, yet.
Transferability		The generality of the hyPDEVS modeling approach allows it to be applied to different physical domains, industries as well as research questions. It is however not without effort.
Separation of modeling and simulation		The standard abstract DEVS simulation engine is tightly coupled with its models. There are, however, other mechanisms to provide separation on the implementation level, for example inheritance in object-oriented programming.

Aside from the usability issues described in table 4.5, one of the main advantages of the hyPDEVS approach that has emerged during the evaluation is the possibility for an integrated description of continuous and discrete modeling aspects on the atomic level, which results in better modularity, maintainability and ultimately also model reusability.

Table 4.7: Evaluation of simulation performance for the DEVS-based modeling approach.

Criterion		Assessment
Performance efficiency	●	The hyPDEVS approach allows for a lean and performance-efficient implementation with little overhead for communication. Moreover, there are techniques to improve simulation efficiency, for example by flattening the hierarchical model structure during compilation [20] to avoid passing simulator messages through multiple hierarchy levels.
Accuracy	●	Because of the tighter integration of continuous and discrete simulation aspects, additional numerical errors for example from extrapolating signals are avoided, thereby improving accuracy. Discrete events can be processed and propagated immediately without delay.
Reliability	◐	Increased modeling transparency for the user also reduces instances where unexpected behaviour occurs. Depending on the software implementation, reliability is typically high and easier to manage for a single integrated simulation environment.
Use of established algorithms	◐	As proven with the ODE wrapper method (see section 3.1.2), hybrid DEVS-based modeling can be combined with using common and established ODE solvers [27]. For discrete-event simulation on the other hand, a DEVS-specific engine has to be used, not all of which can be considered established.
Data handling	◐	An integrated simulation environment makes it easier to provide a unified infrastructure for managing simulation results, model parameters or scenario configurations. Still, handling heterogeneous data, like for example continuous time series together with discrete events, requires some effort.
Distributed simulation	◐	Although hyPDEVS is based on Parallel DEVS and thus facilitates parallel and distributed simulation, it usually involves significant communication between the continuous solver and discrete-event simulation engine. Therefore, the performance efficiency in a distributed simulation depends on how the model is partitioned.

































4.6 Comparison

After assessing the co-simulation and DEVS-based modeling approach individually, we now provide a direct side-by-side comparison with regard to the evaluation criteria, shown in table 4.8. From the comparison, one can see that, generally speaking, co-simulation demands less effort for model development, due to the broad support from

















4. EVALUATION AND COMPARISON

different established simulators, available component libraries or model import/export functionality. On the other hand, subsequent maintainability, performance efficiency and ease of use for non-experts are lower, partly due to the modelling restrictions posed by the co-simulation architecture.

Table 4.8: Comparison of evaluation criteria between co-simulation and DEVS-based modeling.

Criterion	Co-simulation	DEVS-based approach
Ease of use: Model development (expert)		
Ease of use: Simulation user (non-expert)		
Ease of learning		
Ease of debugging		
Modeling transparency		
Collaboration		
General features		
Time effort for model building		
Hierarchical model building		
Modularity		
Model reusability		
Flexibility		
Libraries with pre-defined components		
Level of detail		
Maintainability		
Model import/export		

(continued)

Transferability		
Separation of modeling and simulation		
Performance efficiency		
Accuracy		
Reliability		
Use of established algorithms		
Data handling		
Distributed simulation		

4.7 Conclusion

As mentioned before, we do not intend to judge the importance of individual evaluation criteria, as this depends on the concrete application scenario. For some cases, it might be sufficient to provide a low-effort initial implementation for a short-term simulation study without intention of subsequent reuse. Then a co-simulation approach might be a more suitable choice. On the other hand, if someone aims to develop a simulation environment with integrated hybrid modeling capabilities and long-term use for non-experts, then it might pay off to invest a higher initial development effort of a DEVS-based approach in order to gain benefits in the long run.

The goal of this thesis was to identify whether a hybrid DEVS-based approach to modeling and simulation of interdisciplinary production systems can be a suitable choice compared to a common co-simulation approach. As presented in the case study and subsequent evaluation, co-simulation presents a solution to hybrid simulation with lower initial effort for model development. We were able to confirm the core advantages of the co-simulation approach promised prior to the case study (see page 8), namely convenient modeling, suited simulation algorithms and simultaneous model engineering. However, managing simulator coupling and data exchange still presents a challenge and individual model developers have to communicate closely and agree on common interfaces. This includes not only an agreement on which variables need to be exchanged, but especially also their semantics (e.g. order of variables in a vector, units of measurement), since semantics often gets lost during low-level data exchange from one simulator to another. Here, a common reference model can support communication between experts from different fields, thereby reducing misunderstandings and subsequent mistakes.

The formalism-based approach to modeling using hyPDEVS is general in nature, thereby demanding a lot of effort for initial model development, because specialized and domain-specific features (e.g. robust entity exchange, higher-level communication between model components, specialized modelling details) first need to be developed by simulation experts. Once these features are developed and supported by the simulation environment, it becomes easier for other model developers to implement similar specialized models. For long-term applications, the DEVS-based approach can enable significant modeling and simulation advantages, even compared to co-simulation, including model maintainability and extendibility, ease of use for non-expert simulation users, modularity and consequently the ability to develop libraries of hybrid model components for others to use. In addition, hyPDEVS as a specification is open and documented, thereby encouraging others to share their implementations and develop similar software without having to rely on third-party or proprietary software.

The reason the DEVS-based approach presents significant advantages for hybrid modeling lies in the fact that hyPDEVS is able to provide integration of continuous and discrete modeling aspects on the model level, as illustrated in figure 4.2. Compared to co-simulation, which achieves integration on the simulation/runtime level by exchanging data, integration on modeling level unifies hybrid model development, thereby eliminating restrictions regarding model decomposition and modularity.

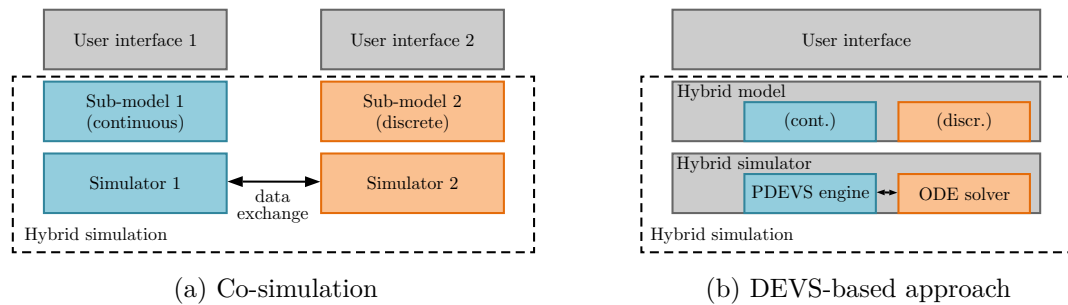


Figure 4.2: Abstract three-tier architecture of a simulation environment (including user interface, model and simulation engine) showing different levels of integration for co-simulation and DEVS-based approach. For co-simulation, integration is typically achieved only on the data simulation level by exchanging data during runtime. The DEVS-based approach on the other hand allows integration of hybrid modeling aspects also on the modeling level, thereby providing better modularity of hybrid components as well as a consistent user interface.

Although some studies suggest that there are modeling restrictions and shortcomings of Parallel DEVS (on which hyPDEVS is based), see [102], [103] for example, these did not pose any substantial restrictions in our application and could easily be managed by adequate modeling.

To summarize, and coming back to the research questions specified at the beginning of this work (see section 1.2), we can conclude:

Research Question 1: *Is a formal modeling approach based on DEVS suited for interdisciplinary modeling and simulation of production systems?*

The answer to this question is yes. Choosing especially a hybrid DEVS-based formalism like hyPDEVS allows to model continuous as well as discrete characteristics in an integrated manner in order to incorporate all relevant aspects for interdisciplinary investigations of production systems. This includes discrete entities for logistics simulation, state machines for event-driven logic (processing states, information messages, etc.) and continuous dynamics in terms of differential equations. Especially differential equations make it possible for example to model energetically relevant transient processes with sufficient level of detail (compared to a purely discrete modeling approach, for example) for comprehensive analysis of energy efficiency in an industrial context. The formal approach makes little restrictions on what can be modelled in principle. As some of the main advantages, a DEVS-based approach greatly facilitates modularity (even of hybrid components), hierarchical model building (due to its *closure under coupling*) and, as a result, reusability of hybrid models.

However, we also demonstrated in this thesis that the hyPDEVS formalism is generic an basic in nature. For more specialized modeling applications (e.g. industrial physical systems), model developers have to make effort to implement higher-level modeling features themselves, like robust exchange of production entities for example.

Research Question 2: *How does the DEVS-based approach compare to a common co-simulation solution with regard to interdisciplinary production systems?*

Unlike co-simulation, where integration is achieved in terms of data exchange on the simulation/runtime level, the hyPDEVS approach provides integration of all relevant modeling aspects (especially hybrid) also on the *modeling level* (see figure 4.2). This presents one of the main differences between these two concepts. The tighter integration enables several potential benefits, especially from a user perspective, including more fine-grained modularity (even for hybrid components) and improved reusability. Many of the features possible with co-simulation (distributed simulation, level of detail, etc.) are also possible in principle using a DEVS-based approach. To encourage its use, however, extensions to support high-level modeling features need to be developed.

Summary & Outlook

5.1 Summary

In this work, we presented two approaches for hybrid modeling in the context of interdisciplinary simulation of industrial systems. *Co-simulation* aims at coupling multiple simulators to form a single hybrid simulation. In comparison, the presented *DEVS-based* approach provides a formal description of hybrid components (atomics) as well as coupled models. For each of these two modeling approaches, we presented a case study on the simulation of a production facility involving different engineering domains, such as production machines, logistics, energy systems and thermal building aspects. The first case study was developed using BCVTB as common co-simulation middleware that handles synchronization and data exchange between multiple simulator clients. Communication interfaces were derived based on a common *reference model*. For the second case study, we employed a modular approach, where the production facility is divided into well-defined components, called *cubes*.

Based on the experiences gathered during the case studies, we then evaluated and compared both modeling approaches using criteria derived from relevant literature. The qualitative results of the evaluation make no claim to be exhaustive, but rather aim at providing an understanding of which approach is better suited for a particular simulation application.

It could be confirmed that co-simulation enables to employ established model descriptions and suitable simulation algorithms on the one hand, and allows distributed model development by multiple domain experts on the other hand. However, the co-simulation case study also showed the importance of a shared reference model, in order for all model developers to agree on common semantics of the co-simulation interfaces when using low-level data exchange.

Compared to co-simulation, the DEVS-based approach is able to provide integration of continuous and discrete modeling aspects not just on the data level, but also on the

modeling level, which entails several major benefits for model development, including improved modularity of hybrid components, model maintainability and ultimately better reusability. However, some of the presented disadvantages of DEVS-based modeling concern the genericity of the approach, which does not provide high-level support for specialized modeling features, such as robust exchange of entities or specialized information interfaces.

The benefits of the DEVS-based hybrid approach may potentially incentivise simulation software developers to overcome the obstacle of higher initial demand for model development in order to develop hybrid simulation tools based on hyPDEVS. Once higher-level modeling features are supported by simulation software, DEVS-based hybrid simulation could gain more widespread acceptance in the industry.

5.2 Future Work

One open topic that has not been fully addressed in this work is the validation and comparison of simulation accuracy of the hyPDEVS solution compared to other established implementations. Although a basic validation has been carried out in order to ensure basic consistency, further studies need to investigate in greater depth the potential discrepancies of this approach.

Based on the work presented in this thesis, several potential directions for future work become visible. Regarding future development of co-simulation, the task of coupling simulation tools could be greatly improved by providing *semantic information coupling* as oppose to low-level data exchange. There are some studies that aim at employing system integration technologies for co-simulation, for example [84] presents an approach based on OPC Unified Architecture (OPC UA). However, these solutions are still in their infancy and further developments could potentially eliminate some of the current drawbacks of co-simulation while at the same time keeping its major advantages.

With regard to DEVS-based hybrid modeling, it was already mentioned that it would be important to develop higher-level modeling features for specific domains of application. These features could be provided as a kind of *intermediate modeling layer* that still uses hyPDEVS as a substructure, but provides an automatic translation from a high-level model description to a hyPDEVS-compliant implementation. This intermediate layer would provide an abstraction of low-level DEVS functions, thereby hiding the specific formalism from the user. DEVS-based simulation tools could potentially develop different intermediate modeling layers for different fields of application while keeping a common basis that would hold open the door for multi-domain modeling considerations.

MatlabDEVS Source Code for the Oven Model

```
%% Ofen_v2_Cube
%%
classdef Ofen_v2_cube < hybridatomic
    %% Description
    % Class definition file for a *hybrid atomic PDEVS model
    % that implements an oven for BaMa MOBA Example
    %[...]
    properties (Access = public)
    end
    methods
        function obj = Ofen_v2_cube(name,inistates,c_inistates,
            parameters,elapsed)
            global Aplan
            if nargin == 5
                x = {'EIN','EOUTcom','EAcom','Pplan'};
                y = {'EOUT','EA','EINcom','state'};
            s = {'sigma','p','h','last_p','ent','enttype','entBuff','
                entBuffGt','count','log','ACK','step','event','EOUTsent','
                EAsent'};
            sysparams = struct('N',num2str(parameters.N),'P_H',
                num2str(parameters.P_H),'P_S',...
                num2str(parameters.P_S),'tB',num2str(parameters.
                    tB),'Tsoll',num2str(parameters.Tsoll),'H',
                    num2str(parameters.H),...
                    'V',num2str(parameters.V),'UA',num2str(parameters
                        .UA),'cpL',num2str(parameters.cpL),'rhoL',
                        num2str(parameters.rhoL),...
                        'eta',num2str(parameters.eta),'alpha',num2str(
                            parameters.alpha),'sign',parameters.sign);
```

```
        c_states = zeros(1,4);
        mealy = 1;
    else
        error('mistake at constructor method for class
            Ofen_v2_cube');
    end
    obj = obj@hybridatomic(name,x,y,s,c_states,mealy,elapsed,
        sysparams); % incarnate the associated hybrid
        simulator
    obj.output_length = 4;

% initialize the continuous states
    obj.c_states = c_inistates';

% initialize the discrete states
    obj.s.sigma = inistates.sigma;
    obj.s.p = inistates.p;
    obj.s.h = inistates.h;
    obj.s.ent = cell(1,eval(obj.sysparams.N));
    obj.s.enttype = 0; %necessary for accessing Aplan
    obj.s.entBuff = Entity.empty();
    obj.s.entBuffGt = -1;
    obj.s.count = inistates.count;
    obj.s.ACK = 0;
    obj.s.last_p = '';
    obj.s.step = eval(obj.sysparams.tB)/eval(obj.sysparams.N)
    obj.s.event = false;
    obj.s.EOUTsent = false;
    obj.s.EAsent = false;
end

% Time advance ta
    function ta = tafun(obj)
        ta = obj.s.sigma;
    end

% Confluent transition function delta_conf
    function deltaconffun(obj,gt)
    printLog('t=%.2f: [dconf] %s\n',gt, obj.name);
        deltaintfun(obj);
        deltaextfun(obj,gt);
    end

% External transition function delta_ext
    function deltaextfun(obj,gt)
        global Aplan
        %incoming Pplan signal
        if ~isempty(obj.x.Pplan) && ...
            (strcmp(obj.s.p, 'standby') || strcmp(obj.s.p, '

```

```

        off') || ...
        strcmp(obj.s.p, 'holding') || strcmp(obj.s.p,
        'heating') || strcmp(obj.s.p, 'waiting'))
if obj.x.Pplan{1} <= 0.5
    obj.s.p = 'off';
elseif obj.x.Pplan{1} <= 1.5
    obj.s.p = 'standby';
else    %switch to heating or immediately to waiting,
        depending on current temperature (c_states(1))
    if obj.sysparams.sign*(obj.c_states(1) - eval(obj
        .sysparams.Tsoll)) >= 0
        obj.s.p = 'waiting';
    else
        obj.s.p = 'heating';
    end
end
obj.s.sigma = 0;    %schedule sending state update
printLog('t=%.2f: [dext] %s receiving Pplan signal,
        switching to state %s\n',gt, obj.name, obj.s.p);
%enttype
if ~isempty(obj.x.Pplan{2})
    obj.s.enttype = obj.x.Pplan{2};
    printLog('t=%.2f: [dext] %s receiving entity type
        %d from Pplan\n',gt, obj.name, obj.s.enttype)
    ;
end

elseif ~isempty(obj.x.Pplan)
    if obj.tnext > gt && obj.s.sigma > 0
        obj.s.sigma = obj.tnext - gt;
    end
    printLog('t=%.2f: [dext] %s ignoring Pplan signal\n',
        gt, obj.name);
end

%incoming EOUTcom, EAcum need to be checked first, to
make room on the belt
if ~isempty(obj.x.EOUTcom)
    obj.s.EOUTsent = true;
    printLog('t=%.2f: [dext] %s receiving EOUTcom\n',gt,
        obj.name);
end
if ~isempty(obj.x.EAcum)
    obj.s.EAsent = true;
    printLog('t=%.2f: [dext] %s receiving EAcum\n',gt,
        obj.name);
end
if obj.s.EOUTsent && obj.s.EAsent
    printLog('t=%.2f: [dext] %s deleting entity and

```

```
        shifting\n',gt, obj.name);
%shift entities, update counters
shift_belt(obj);

%switch to state waiting or holding
%note: state might be overwritten by incoming entity
(see below)
if sum(cellfun('isempty',obj.s.ent)) == eval(obj.
sysparams.N)
    obj.s.p = 'waiting';
    obj.s.sigma = inf;
else
    obj.s.p = 'holding';
    obj.s.sigma = obj.s.step;
end
obj.s.EOUTsent = false;
obj.s.EAsent = false;
end

% reset buffer if not same timestep (where buffer was
filled)
if obj.s.entBuffGt ~= gt && ~isempty(obj.s.entBuff)
    obj.s.entBuff = Entity.empty();
    printLog('t=%.2f: [dext] %s clearing buffer\n',gt,
obj.name);
end
if ~isempty(obj.x.EIN)
    printLog('t=%.2f: [dext] %s putting entity (id=%d) in
buffer\n',gt, obj.name, obj.x.EIN(1).id);
    obj.s.entBuff = obj.x.EIN;
end

%incoming entity on port EIN
if ~isempty(obj.s.entBuff) && (isempty(obj.s.ent) ||
isempty(obj.s.ent{1})) && ...
    (strcmp(obj.s.p, 'waiting') || strcmp(obj.s.p, '
holding'))
    %add entity to list and consume
    obj.s.ent{1} = obj.s.entBuff;
    obj.s.entBuff = Entity.empty();
    obj.s.count = obj.s.count + 1;
    obj.s.p = 'holding';

    obj.s.sigma = 0;    %schedule sending ACK
    obj.s.ACK = true;
    printLog('t=%.2f: [dext] %s accepting entity from
buffer, sigma=%.2f\n',gt, obj.name,obj.s.sigma);

elseif ~isempty(obj.s.entBuff)
```

```

        if obj.tnext > gt && obj.s.sigma > 0
            obj.s.sigma = obj.tnext - gt;
        end
        if obj.s.entBuffGt ~= gt
            obj.s.entBuffGt = gt;
            printLog('t=%.2f: [dext] %s saving buffer time,
                    sigma=%.2f\n',gt, obj.name, obj.s.sigma);
        end
    end
end

% Internal transition function delta_int
function deltaintfun(obj)
    global Aplan
    printLog('t=%.2f: [dint] %s in %s\n', obj.tnext, obj.name, obj.
s.p);
    obj.s.event = false;
    %if sigma not set explicitly further down, go passive
    obj.s.sigma = inf;

    %finished sending ACK
    if obj.s.ACK == true
        obj.s.ACK = false;
        obj.s.sigma = obj.s.step;
        printLog('t=%.2f: [dint] %s ACK was sent, sigma=%.2f\
n', obj.tnext, obj.name, obj.s.sigma);
        return
    end
    if strcmp(obj.s.p, 'holding')
        % if there is a entity waiting on the furthest spot
        if ~isempty(obj.s.ent{eval(obj.sysparams.N)})
            obj.s.p = 'output';
            obj.s.sigma = 0;
            printLog('t=%.2f: [dint] %s furthest spot filled\
n', obj.tnext, obj.name);
        else
            % shift immediately and stay in holding
            shift_belt(obj)
            obj.s.sigma = obj.s.step;
        end
        printLog('t=%.2f: [dint] %s in holding, sigma=%.2f\n'
, obj.tnext, obj.name, obj.s.sigma);
        return
    elseif strcmp(obj.s.p, 'output')
        % stay in output (furthest spot blocked) and
        reschedule
        obj.s.sigma = obj.s.step;
        printLog('t=%.2f: [dint] %s in output, sigma=%.2f\n',
            obj.tnext, obj.name, obj.s.sigma);
    end
end

```

```
        return
    end

end

% Discrete output function lambda
function lambdafun(obj)
    global Aplan
    %send state
    if ~strcmp(obj.s.last_p, obj.s.p)
        %printLog('t=%.2f: [lambda] %s sending state=%s\n',
            obj.tnext, obj.name, obj.s.p);
        obj.y.state = obj.s.p;
        obj.s.last_p = obj.s.p;
    end

    %send ACK
    if obj.s.ACK == true
        obj.y.EINcom = 1;
        printLog('t=%.2f: [lambda] %s sending ACK\n', obj.
            tnext, obj.name);
    end

    %try to sent entities on EOUT and EA
    if strcmp(obj.s.p, 'output')
        %send EOUT
        if ~obj.s.EOUTsent
            obj.y.EOUT = obj.s.ent{eval(obj.sysparams.N)};
            obj.y.EOUT.m = obj.y.EOUT.m * (1-eval(obj.
                sysparams.alpha));
            printLog('t=%.2f: [lambda] %s sending entity (id
                =%d) on EOUT\n', obj.tnext, obj.name, obj.y.
                EOUT.id);
        end
        %send EA
        if ~obj.s.EAsent && eval(obj.sysparams.alpha) > 0
            obj.y.EA = obj.s.ent{eval(obj.sysparams.N)};
            obj.y.EA.m = obj.y.EA.m * eval(obj.sysparams.
                alpha);
            printLog('t=%.2f: [lambda] %s sending entity (id
                =%d) on EA\n', obj.tnext, obj.name, obj.y.EA.
                id);
        else
            %alpha = 0 -> EA does not need to be sent
            obj.s.EAsent = true;
        end
    end
end

% Rate of change function f
```

```

function dq = f(obj,gt,x,y)
    global Aplan
    % calculate sum of all entities' influence (m*cp)
    ent_sum = 0;
    for i=1:eval(obj.sysparams.N)
        if ~isempty(obj.s.ent{i})
            ent_sum = ent_sum + obj.s.ent{i}.m*obj.s.ent{i}.
                cp;
        end
    end

    % differential equation: oven temperature T
    dq(1) = (x(2)-(y(1)-x(3))*eval(obj.sysparams.UA))/...
        (eval(obj.sysparams.cpL)*eval(obj.sysparams.rhoL)
            *eval(obj.sysparams.V) + ent_sum);

    % dummies for remaining continuous states
    dq(2:4) = zeros(1,3);

    %set temperature of next entity to be sent to oven
    temperature
    if ~isempty(obj.s.ent{end})
        obj.s.ent{end}.T = y(1);
    end
    obj.c_states=y';
end

% State event condition function c_se
function ret = cse(obj,gt,y)
    global Aplan
    %state event for when desired temperature is reached
    %direction of state event corresponds to sign parameter (
        oven vs. freezer)
    if ~obj.s.event && strcmp(obj.s.p, 'heating')
        ret = [y(1)-eval(obj.sysparams.Tsoll), obj.sysparams.
            sign, 1];
    else
        ret = [0, obj.sysparams.sign*1, 1];
    end
end

% State event transition function delta_state
function obj = deltastatefun(obj,gt,y,event_number)
    global Aplan
    printLog('t=%.2f: [dsf] %s entering deltastatefun\n', gt,
        obj.name);
    obj.s.event = true;

    % desired temperature reached

```

```
if event_number == 1
    if strcmp(obj.s.p, 'heating')
        obj.s.p = 'waiting';
        obj.s.sigma = 0;
        printLog('t=%.2f: [dsf] %s temperature reached,
            switching to state %s\n', gt, obj.name, obj.s.
            p);
    end
end
end

% Continuous output function lambda_c
function cy = lambda_c(obj,gt,y,x)
    % inputs: x(1) ... Pel
    %          x(2) ... Qw
    %          x(3) ... Tu
    % states: y(1) ... T
    % outputs: cy(1) ... QAW
    %           cy(2) ... Qrec
    %           cy(3) ... PelB
    %           cy(4) ... QwB
    global Aplan
    printLog('t=%.2f: [lambda_c] %s entering lambda_c\n', gt,
        obj.name);
    cy = zeros(4,1);

    %algebraic equation: waste heat Q_WH
    cy(1) = ((y(1)-x(3))*eval(obj.sysparams.UA)+x(1))*(1-eval(
        obj.sysparams.eta));

    %algebraic equation: recoverable heat Q_rec
    cy(2) = ((y(1)-x(3))*eval(obj.sysparams.UA)+x(1))*eval(
        obj.sysparams.eta);

    %PelB and QwB:
    switch obj.s.p
        case 'off'
            cy(3) = 0;
            cy(4) = 0;
        case 'standby'
            cy(3) = eval(obj.sysparams.P_S);
            cy(4) = 0;
        otherwise %states 'heating','waiting','holding','
            shifting'
            cy(3) = eval(obj.sysparams.P_S);
            cy(4) = obj.sysparams.sign*controller(obj.
                sysparams.sign*(eval(obj.sysparams.Tsoll)-y(1)
                ),obj);
    end
end
```

```

        %logging for plotting of simulation results
        %[...]
    end

    % Auxiliary function for two-level controller
    function dQ = controller(dT,obj)
        if strcmp(obj.s.h,'on')
            if dT<-eval(obj.sysparams.H)
                obj.s.h='off';
            end
        else
            if dT>eval(obj.sysparams.H)
                obj.s.h='on';
            end
        end
        if strcmp(obj.s.h,'on')
            dQ=eval(obj.sysparams.P_H);
        else
            dQ=0;
        end
    end
end

% Auxiliary function for shifting entities
function shift_belt(obj)
    printLog('t=?: [%s] %s shifting belt\n', obj.s.p, obj.
        name);
    obj.s.ent = {[], obj.s.ent{1:length(obj.s.ent)-1}};
end
end
end

```


List of Figures

2.1	Different coupling strategies for co-simulation (adapted from [114] and [113]). The blue and red arrows denote computation steps in simulator 1 and simulator 2, respectively. The black arrows denotes data exchange.	9
2.2	Data exchange between two coupled simulators according to Jacobi type of loose coupling. Both input variables \tilde{u}_1^k and \tilde{u}_2^k have to be extrapolated between communication intervals.	12
2.3	Data exchange between two coupled simulators according to Gauß-Seidel type of loose coupling. Simulator 1 is executed first and has to extrapolate its input \tilde{u}_1^k , simulator 2 can use interpolation for \tilde{u}_2^k	12
2.4	Reference model overview (taken from [71]). Physical components are in blue, information components in red, plan elements in green. Black arrows show dynamic variable connections, green arrows are static parameter associations.	15
2.5	Communication and data exchange between simulation tools is managed via middleware.	17
2.6	Architecture overview of the BCVTB software (adapted from [139]).	18
2.7	Overall framework for co-simulation between MATLAB, Dymola and EnergyPlus. A second MATLAB instance provides processing and visualization of simulation results.	19
2.8	Graphical user interface in MATLAB for executing the simulation and plotting simulation results.	19
2.9	Multi-domain object diagram of a powertrain from the main drive of a turning lathe (taken from [48]). The diagram models electric energy supply of an asynchronous machine, the conversion to mechanical energy as well as transfer of waste heat.	21
2.10	Power consumption of a machine tool over time (left) with detail (right) showing base load P_{base} , dynamic load P_{dyn} and cutting power P_{cut} (adapted from [50]).	22
2.11	BCVTB graphical user interface showing three actors (responsible for communicating with the simulator clients) and signal connections.	25
2.12	Overall co-simulation framework instance for the case study with BCVTB middleware as well as MATLAB, Dymola and EnergyPlus simulation clients and post-processing in MATLAB.	29

2.13	Comparison of annual final energy demand for the three scenarios. Scenario 1 shows demand for natural gas to operate the CHP unit.	30
2.14	Detailed energy flow for the energy system configuration of scenario 1 (adapted from [71]). The analysis shows that the high gas demand is a results from inefficient operation of the absorption chillers.	30
3.1	Operating principle of an atomic DEVS (taken from [102]).	32
3.2	Input/Output events and state trajectory for an atomic DEVS model (adapted from [69]).	33
3.3	Example configuration of a production facility consisting of different cubes. .	41
3.4	Different categories of cubes, divided into four areas: Machine and production process, building, energy system and technical building services, and logistics.	42
3.5	Hybrid nature of cube models encapsulating discrete material flow and continuous energy behaviour.	43
3.6	Interface of the presented oven model showing inputs (left) and outputs (right). The figure also shows parameters and state variables of the internal model. .	44
3.7	State diagram describing the discrete behaviour of the oven cube.	45
3.8	Simple example of two stations exchanging an entity, either using an acknowledgement signal (ACK) or request signal (REQ).	47
3.9	Example of a simple production facility, consisting of a processing line (top), energy system (bottom left) and thermal building cubes.	49
3.10	Comparison of entity flow: Number of entities over time in the different stations.	52
3.11	Comparison of energy demand: Energy consumption for heating, cooling and electric energy over time.	52
3.12	Comparison of oven and freezer operation: Temperature profile for oven and freezer over time.	53
3.13	Comparison of thermal zone temperatures: Temperature profile for thermal zones as well as ambient temperature over time.	53
4.1	Five-point scale for assessment of evaluation criteria.	58
4.2	Abstract three-tier architecture of a simulation environment (including user interface, model and simulation engine) showing different levels of integration for co-simulation and DEVS-based approach. For co-simulation, integration is typically achieved only on the data simulation level by exchanging data during runtime. The DEVS-based approach on the other hand allows integration of hybrid modeling aspects also on the modeling level, thereby providing better modularity of hybrid components as well as a consistent user interface. . . .	72

List of Tables

2.1	Classification of methods for coupled simulation (adapted from [113] and [135])	6
2.2	Overview of the simulation scenarios comparing different design variants for the energy system	27
3.1	Production schedules for two example scenarios	51
4.1	Evaluation criteria including qualitative scales, grouped into user considerations, modeling capabilities, and simulation performance.	59
4.2	Evaluation of user consideration for the co-simulation modeling approach. . .	62
4.3	Evaluation of modeling capabilities for the co-simulation modeling approach.	63
4.4	Evaluation of simulation performance for the co-simulation modeling approach.	65
4.5	Evaluation of user consideration for the DEVS-based modeling approach. . .	66
4.7	Evaluation of simulation performance for the DEVS-based modeling approach.	69
4.8	Comparison of evaluation criteria between co-simulation and DEVS-based modeling.	70

Bibliography

- [1] A. Arisha and M. El Baradie, “On the Selection of Simulation Software for Manufacturing Application”, in *Conference Papers*, Queen’s University Belfast, N. Ireland, Aug. 28–30, 2002, pp. 495–507. [Online]. Available: <http://arrow.dit.ie/buschmarcon/91>.
- [2] M. Arnold, A. Carrarini, A. Heckmann, and G. Hippmann, “Simulation Techniques for Multidisciplinary Problems in Vehicle System Dynamics”, in *Vehicle System Dynamics Supplement 40*, M. Valasek, Ed., Vienna, Austria, 2004, pp. 17–36, ISBN: 978-90-265-1970-3. [Online]. Available: <http://elib.dlr.de/12231/> (visited on Oct. 27, 2016).
- [3] M. U. Awais, “Distributed hybrid co-simulation”, Dissertation, TU Wien, Wien, 2015. [Online]. Available: <http://media.obvsg.at/p-AC12706426-2001> (visited on Oct. 24, 2016).
- [4] Z. Ayağ, “Evaluating simulation software alternatives through ANP”, in *Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia*, 2011. [Online]. Available: <http://ieomsociety.org/ieom2011/pdfs/IEOM079.pdf> (visited on Nov. 4, 2016).
- [5] B. Bailey, R. Klein, and S. Leef, “Hardware/Software co-simulation strategies for the future”, *Mentor Graphics Co.*, <http://www.mentor.com>, 2000. [Online]. Available: http://newit.gsu.by/resources/articles/Mentor%5Ccosim_strategies.pdf (visited on Oct. 7, 2016).
- [6] O. Balci, “A life cycle for modeling and simulation”, *SIMULATION*, vol. 88, no. 7, pp. 870–883, Jul. 1, 2012, ISSN: 0037-5497, 1741-3133. DOI: 10.1177/0037549712438469. [Online]. Available: <http://sim.sagepub.com/cgi/doi/10.1177/0037549712438469> (visited on Oct. 31, 2016).
- [7] O. Balci, J. D. Arthur, and R. E. Nance, “Accomplishing reuse with a simulation conceptual model”, in *2008 Winter Simulation Conference*, Dec. 2008, pp. 959–965. DOI: 10.1109/WSC.2008.4736162.

- [8] J. Banks, “Selecting Simulation Software”, in *Proceedings of the 23rd Conference on Winter Simulation*, ser. WSC ’91, Washington, DC, USA: IEEE Computer Society, 1991, pp. 15–20, ISBN: 978-0-7803-0181-8. [Online]. Available: <http://dl.acm.org/citation.cfm?id=304238.304249> (visited on Nov. 2, 2016).
- [9] F. J. Barros, “Dynamic structure discrete event system specification: A new formalism for dynamic structure modeling and simulation”, in *Proceedings of the 27th Conference on Winter Simulation*, IEEE Computer Society, 1995, pp. 781–785. [Online]. Available: <http://dl.acm.org/citation.cfm?id=224731> (visited on Oct. 30, 2016).
- [10] J. Bastian, C. Clauß, S. Wolf, and P. Schneider, “Master for co-simulation using FMI”, in *Proceedings of the 8th International Modelica Conference; March 20th–22nd; Technical University; Dresden; Germany*, Linköping University Electronic Press, 2011, pp. 115–120. [Online]. Available: <http://www.ep.liu.se/ecp/article.asp?issue=63&volume=&article=14> (visited on Nov. 5, 2016).
- [11] F. Bergero and E. Kofman, “PowerDEVS: A tool for hybrid system modeling and real-time simulation”, *SIMULATION*, vol. 87, pp. 113–132, 1-2 Jan. 1, 2011, ISSN: 0037-5497, 1741-3133. DOI: 10.1177/0037549710368029. [Online]. Available: <http://sim.sagepub.com/cgi/doi/10.1177/0037549710368029> (visited on Oct. 31, 2016).
- [12] F. Bleicher, F. Duer, I. Leobner, I. Kovacic, B. Heinzl, and W. Kastner, “Co-simulation environment for optimizing energy efficiency in production systems”, *CIRP Annals - Manufacturing Technology*, vol. 63, no. 1, pp. 441–444, 2014, ISSN: 0007-8506. DOI: 10.1016/j.cirp.2014.03.122. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0007850614001255> (visited on Sep. 26, 2016).
- [13] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Jungmanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J. V. Peetz, and S. Wolf, “Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models”, *8th International Modelica Conference 2011*, pp. 173–184, Nov. 2009. DOI: 10.3384/ecp12076173. [Online]. Available: <http://www.ep.liu.se/ecp/076/017/ecp12076017.pdf>.
- [14] E. Bonneville and A. Rialhe, “Good practice for energy efficiency in industry”, *Efficiency & Ecodesign*, 2006. [Online]. Available: <http://www.leonardo-energy.info/sites/leonardo-energy/files/root/Documents/2009/DSM-industry.pdf> (visited on Nov. 10, 2016).
- [15] V. Bosilj-Vuksic, V. Ceric, and V. Hlupic, “Criteria for the evaluation of business process simulation tools”, *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 2, pp. 73–88, 2007, ISSN: 1555-1229. [Online]. Available: <http://www.ijikm.org/Volume2/IJIKMv2p073-088Bosilj396.pdf> (visited on Nov. 2, 2016).

- [16] F. Bouchhima, M. Briere, G. Nicolescu, M. Abid, and E. M. Aboulhamid, “A SystemC/Simulink co-simulation framework for continuous/discrete-events simulation”, in *2006 IEEE International Behavioral Modeling and Simulation Workshop*, IEEE, 2006, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4062043 (visited on Oct. 28, 2016).
- [17] M. Busch, M. Arnold, A. Heckmann, and S. Dronka, “Interfacing SIMPACK to Modelica/Dymola for multi-domain vehicle system simulations”, *SIMPACK News*, vol. 11, no. 2, pp. 1–3, 2007. [Online]. Available: http://simpack.com/fileadmin/simpack/doc/newsletter/2007/SN-2-2007_n.pdf (visited on Oct. 29, 2016).
- [18] F. E. Cellier and E. Kofman, *Continuous System Simulation*. New York: Springer, 2006, 643 pp., ISBN: 978-0-387-26102-7 978-0-387-30260-7.
- [19] H. Cho and Y. Cho, “Devs-C++ Reference Guide”, *The University of Arizona*, vol. 7, p. 11, 1997. [Online]. Available: <http://acims.asu.edu/wp-content/uploads/2012/02/devsc-user-ref.pdf> (visited on Oct. 31, 2016).
- [20] A. C. H. Chow and B. P. Zeigler, “Parallel DEVS: A Parallel, Hierarchical, Modular, Modeling Formalism”, in *Proceedings of the 1994 Winter Simulation Conference*, ser. WSC ’94, San Diego, CA, USA: Society for Computer Simulation International, 1994, pp. 716–722, ISBN: 978-0-7803-2109-0. [Online]. Available: <http://dl.acm.org/citation.cfm?id=193201.194336> (visited on Oct. 30, 2016).
- [21] S. Ciraci, J. Daily, and J. Fuller, “FNCS: A framework for power system and communication networks co-simulation”, *Proceedings Symposium on Theory of Modeling & Simulation-DEVS Integrative*, vol. 46, no. 4, pp. 256–263, 2014, ISSN: 07359276.
- [22] M. C. D’Abreu and G. A. Wainer, “M/CD++: Modeling continuous systems using Modelica and DEVS”, in *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Sep. 2005, pp. 229–236. DOI: 10.1109/MASCOTS.2005.36.
- [23] Dassault Systèmes AB, *Dymola Dynamic Modeling Laboratory: Getting Started with Dymola*, version 2017, Apr. 2016. [Online]. Available: http://www.3ds.com/fileadmin/PRODUCTS/CATIA/DYMOLA/PDF/Getting_started_with_Dymola.pdf (visited on Oct. 8, 2016).
- [24] L. Davis and G. Williams, “Evaluating and Selecting Simulation Software Using the Analytic Hierarchy Process”, *Integrated Manufacturing Systems*, vol. 5, no. 1, pp. 23–32, Mar. 1, 1994, ISSN: 0957-6061. DOI: 10.1108/09576069410050314. [Online]. Available: <http://www.emeraldinsight.com/doi/full/10.1108/09576069410050314> (visited on Nov. 2, 2016).

- [25] E. De Sturler, J. Hoefflinger, L. Kale, and M. Bhandarkar, “A new approach to software integration frameworks for multi-physics simulation codes”, in *The Architecture of Scientific Software*, Springer, 2001, pp. 87–104. [Online]. Available: http://link.springer.com/chapter/10.1007/978-0-387-35407-1_6 (visited on Oct. 28, 2016).
- [26] C. Deatcu and T. Pawletta, “Towards dynamic structure hybrid devs for scientific and technical computing environments”, in *Proceedings MATHMOD 09 Vienna*, Vienna, 2009, pp. 2716–2719, ISBN: 978-3-901608-35-3. [Online]. Available: <http://www.mb.hs-wismar.de/cea/pubs/2009/2009-mathmod-devs.pdf> (visited on Oct. 30, 2016).
- [27] C. Deatcu and T. Pawletta, “A Qualitative Comparison of Two Hybrid DEVS Approaches”, *SNE - Simulation Notes Europe*, vol. 22, no. 1, pp. 15–24, 2012. DOI: 10.11128/sne.22.tn.10107. [Online]. Available: http://www.sne-journal.org/fileadmin/user_upload/tx_pubdb/10107.sne.22.tn_1.pdf.
- [28] C. Eastman, K. Liston, R. Sacks, and K. Liston, *BIM HANDBOOK*, C. M. Eastman, Ed. Hoboken, N.J: Wiley, 2008, 20–21; 65–84; 93–135, ISBN: 978-0-470-18528-5. [Online]. Available: http://s3.amazonaws.com/academia.edu/documents/31207284/BIM_Handbook_1st.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1476630625&Signature=bPQf4Rpas5g05QebDfp/J3UKfhw=&response-content-disposition=inline;%20filename=BIM_handbook_A_guide_to_buildi.
- [29] F. Ebert, “On partitioned simulation of electrical circuits using dynamic iteration methods”, 2008. [Online]. Available: <https://www.depositonce.tu-berlin.de/handle/11303/2294> (visited on Oct. 28, 2016).
- [30] C. Farhat and M. Lesoinne, “Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems”, *Computer Methods in Applied Mechanics and Engineering*, vol. 182, pp. 499–515, 3–4 Feb. 18, 2000, ISSN: 0045-7825. DOI: 10.1016/S0045-7825(99)00206-6. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0045782599002066> (visited on Oct. 28, 2016).
- [31] J. M. Fegan and G. M. Lane, “Introduction to simulation using Intelligent Simulation Interface (ISI)”, in *Simulation Conference, 1991. Proceedings., Winter*, Dec. 1991, pp. 143–147. DOI: 10.1109/WSC.1991.185608.
- [32] C. A. Felippa, K. C. Park, and C. Farhat, “Partitioned analysis of coupled mechanical systems”, *Computer Methods in Applied Mechanics and Engineering*, Advances in Computational Methods for Fluid-Structure Interaction, vol. 190, pp. 3247–3270, 24–25 Mar. 2, 2001, ISSN: 0045-7825. DOI: 10.1016/S0045-7825(00)00391-1. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0045782500003911> (visited on Oct. 28, 2016).

- [33] J.-B. Filippi and P. Bisgambiglia, “JDEVS: An implementation of a DEVS based formal framework for environmental modelling”, *Environmental Modelling & Software*, Concepts, Methods and Applications in Environmental Model Integration, vol. 19, no. 3, pp. 261–274, Mar. 2004, ISSN: 1364-8152. DOI: 10.1016/j.envsoft.2003.08.016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S136481520300210X> (visited on Oct. 31, 2016).
- [34] J. S. Fitzgerald, P. G. Larsen, K. G. Pierce, and M. H. G. Verhoef, “A formal approach to collaborative modelling and co-simulation for embedded systems”, *Mathematical Structures in Computer Science*, vol. 23, pp. 726–750, 04 Aug. 2013, ISSN: 0960-1295, 1469-8072. DOI: 10.1017/S0960129512000242. [Online]. Available: http://www.journals.cambridge.org/abstract_S0960129512000242 (visited on Oct. 28, 2016).
- [35] R. Franceschini, P.-A. Bisgambiglia, L. Touraille, P. Bisgambiglia, and D. Hill, “A survey of modelling and simulation software frameworks using Discrete Event System Specification”, in *OASICS-OpenAccess Series in Informatics*, vol. 43, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014. [Online]. Available: <http://drops.dagstuhl.de/opus/volltexte/2014/4772/> (visited on Oct. 6, 2016).
- [36] R. Franke, F. Casella, M. Otter, M. Sielemann, H. Elmqvist, S. Mattson, and H. Olsson, “Stream Connectors - An Extension of Modelica for Device-Oriented Modeling of Convective Transport Phenomena”, in *Proc. 7th International Modelica Conference*, Oct. 2009, pp. 108–121. DOI: 10.3384/ecp09430078. [Online]. Available: http://www.ep.liu.se/ecp_article/index.en.aspx?issue=043;article=12%20http://www.ep.liu.se/ecp/043/012/ecp09430078.pdf.
- [37] P. A. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. New York: IEEE Press, 2004, 897 pp., ISBN: 978-0-471-47163-9.
- [38] M. Geimer, T. Krüger, and P. Linsel, “Co-Simulation, gekoppelte Simulation oder Simulationskopplung? Ein Versuch der Begriffsvereinheitlichung”, *O+ P Zeitschrift für Fluidtechnik-Aktorik, Steuerelektronik und Sensorik*, vol. 50, pp. 11–12, 2006. [Online]. Available: http://www.fast.kit.edu/mobima/288_486.php (visited on Oct. 26, 2016).
- [39] L. Gheorghe, “Continuous/Discrete Co-simulation interfaces from formalization to implementation”, Dissertation, École Polytechnique de Montréal, 2009. [Online]. Available: <http://publications.polymtl.ca/137/> (visited on Oct. 24, 2016).
- [40] L. Gheorghe, F. Bouchhima, G. Nicolescu, and H. Boucheneb, “Formal definitions of simulation interfaces in a continuous/discrete co-simulation tool”, in *Seventeenth IEEE International Workshop on Rapid System Prototyping (RSP’06)*, IEEE, 2006,

- pp. 186–192. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1630768 (visited on Oct. 7, 2016).
- [41] F. González, M. González, and J. Cuadrado, “Weak coupling of multibody dynamics and block diagram simulation tools”, in *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2009, pp. 93–102. [Online]. Available: <http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1649579> (visited on Oct. 29, 2016).
 - [42] L. Goodyear, Ed., *Qualitative Inquiry in Evaluation: From Theory to Practice*, First edition, ser. Research methods for the social sciences 29, San Francisco, CA: Jossey-Bass, 2014, 295 pp., ISBN: 978-0-470-44767-3.
 - [43] A. Görener, “Comparing AHP and ANP: An application of strategic decisions making in a manufacturing company”, *International Journal of Business and Social Science*, vol. 3, no. 11, 2012. [Online]. Available: <http://search.proquest.com/openview/b737ed6e9120805a2251348721daefc5/1?pq-origsite=gscholar> (visited on Nov. 4, 2016).
 - [44] B. Gu, “Co-Simulation of Algebraically Coupled Dynamic Subsystems”, Dissertation, Massachusetts Institute of Technology, Sep. 2001. [Online]. Available: <http://dspace.mit.edu/bitstream/handle/1721.1/8695/49837078-MIT.pdf?sequence=2> (visited on Oct. 29, 2016).
 - [45] A. Gupta, R. Verma, and K. Singh, “Smart Sim selector: A software for simulation software selection”, *International Journal of Engineering (IJE)*, vol. 3, no. 3, p. 175, 2009. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.227.1790&rep=rep1&type=pdf#page=5> (visited on Nov. 4, 2016).
 - [46] I. Hafner, “Möglichkeiten der co-simulation mit dem building controls virtual test bed für den bereich der objektorientierten modellbildung physikalischer systeme”, Diploma Thesis, TU Wien, Wien, 2013. [Online]. Available: <http://www.ub.tuwien.ac.at/dipl/2013/AC07815180.pdf> (visited on Oct. 24, 2016).
 - [47] A. T. Al-Hammouri, “A comprehensive co-simulation platform for cyber-physical systems”, *Computer Communications*, vol. 36, no. 1, pp. 8–19, Dec. 1, 2012, ISSN: 0140-3664. DOI: 10.1016/j.comcom.2012.01.003.
 - [48] B. Heinzl, “Objektorientierte multi-domain-modellierung und simulation von werkzeugmaschinen”, Diploma Thesis, TU Wien, Wien, 2012.
 - [49] —, “Interdisziplinäre forschung zur energieoptimierung in fertigungsbetrieben”, Blickpunkt Forschung: Energie, TU Wien, Sep. 28, 2015.

- [50] B. Heinzl, W. Kastner, I. Leobner, F. Dür, F. Bleicher, and I. Kovacic, “Using coupled simulation for planning of energy efficient production facilities”, in *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2014 Workshop on*, IEEE, 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6842397 (visited on Oct. 9, 2016).
- [51] B. Heinzl, M. Rößler, N. Popper, I. Leobner, K. Ponweiser, W. Kastner, F. Dür, F. Bleicher, and F. Breiteneker, “Interdisciplinary Strategies for Simulation-Based Optimization of Energy Efficiency in Production Facilities”, in *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, Apr. 2013, pp. 304–309. DOI: 10.1109/UKSim.2013.115. [Online]. Available: <http://ieeexplore.ieee.org/document/6527433/%20http://ieeexplore.ieee.org/ielx7/6527367/6527368/06527433.pdf?tp=&arnumber=6527433&isnumber=6527368%20http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6527433>.
- [52] J. Hensen, M. Bartak, and F. Drkal, “Modeling and simulation of a double-skin facade system/Discussion”, *ASHRAE Transactions*, vol. 108, p. 1251, 2002. [Online]. Available: <http://search.proquest.com/openview/fad16f65b4de69a709594a05254d4a72/1?pq-origsite=gscholar> (visited on Oct. 28, 2016).
- [53] C. Herrmann, S. Thiede, S. Kara, and J. Hesselbach, “Energy oriented simulation of manufacturing systems - Concept and application”, *CIRP Annals - Manufacturing Technology*, vol. 60, no. 1, pp. 45–48, 2011, ISSN: 00078506. DOI: 10.1016/j.cirp.2011.03.127. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0007850611001284>.
- [54] K. Hines, “Pia: A framework for embedded system co-simulation with dynamic communication support”, *Technical Report, University of Washington*, 1996. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.7392&rep=rep1&type=pdf> (visited on Oct. 7, 2016).
- [55] V. Hlupic, “A comparative evaluation of ten manufacturing simulation packages”, *CIT Journal of computing and information technology*, vol. 5, no. 1, pp. 21–32, 1997. [Online]. Available: <http://cat.inist.fr/?aModele=afficheN&cpsidt=2802185> (visited on Nov. 5, 2016).
- [56] V. Hlupic, Z. Irani, and R. J. Paul, “Evaluation framework for simulation software”, *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 5, pp. 366–382, 1999. [Online]. Available: <http://link.springer.com/article/10.1007/s001700050079> (visited on Nov. 3, 2016).
- [57] V. Hlupic, “Simulation modelling software approaches to manufacturing problems”, Dissertation, University of London, 1993. [Online]. Available: <http://core.ac.uk/download/pdf/4187477.pdf> (visited on Nov. 5, 2016).

- [58] ———, “Simulation software selection using SimSelect”, *Simulation*, vol. 69, no. 4, pp. 231–239, 1997. [Online]. Available: <http://sim.sagepub.com/content/69/4/231.short> (visited on Nov. 6, 2016).
- [59] V. Hlupic and R. J. Paul, “Simulation Software in Manufacturing Environments: A Users’ Survey”, *CIT. Journal of Computing and Information Technology*, vol. 1, no. 3, pp. 205–212, Dec. 30, 2015, ISSN: 1846-3908. [Online]. Available: <http://cit.fer.hr/index.php/CIT/article/view/3121> (visited on Nov. 2, 2016).
- [60] K. Holder, “Selecting simulation software: An approach to the problem of selecting software for a given modelling situation”, *OR Insight*, vol. 3, no. 4, pp. 19–24, Oct. 1990, ISSN: 1759-0477. DOI: 10.1057/ori.1990.32. [Online]. Available: <http://link.springer.com/10.1057/ori.1990.32> (visited on Nov. 6, 2016).
- [61] IEEE-SA Standards Board, *IEEE Standard for Modeling and Simulation (M & S) High Level Architecture (HLA): Framework and Rules*. New York: Institute of Electrical and Electronics Engineers, 2010, ISBN: 978-0-7381-6251-5. [Online]. Available: <http://ieeexplore.ieee.org/servlet/opac?punumber=5553438> (visited on Nov. 10, 2016).
- [62] D. Justen, S. Hinzmann, and A. Mubarak, “X-in-the-loop-einsatz des xcp-protokolls”, *ATZe Elektronik*, vol. 5, no. 1, pp. 56–61, Feb. 2010, ISSN: 1862-1791, 2192-8878. DOI: 10.1007/BF03223997. [Online]. Available: <http://link.springer.com/10.1007/BF03223997> (visited on Oct. 28, 2016).
- [63] A. Kandakoglu, M. Celik, and I. Akgun, “A multi-methodological approach for shipping registry selection in maritime transportation industry”, *Mathematical and Computer Modelling*, vol. 49, pp. 586–597, 3–4 Feb. 2009, ISSN: 0895-7177. DOI: 10.1016/j.mcm.2008.09.001. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0895717708003361> (visited on Nov. 6, 2016).
- [64] B. Kaplan and J. A. Maxwell, “Qualitative research methods for evaluating computer information systems”, in *Evaluating the Organizational Impact of Healthcare Information Systems*, Springer, 2005, pp. 30–55. [Online]. Available: http://link.springer.com/chapter/10.1007/0-387-30329-4_2 (visited on Nov. 6, 2016).
- [65] G. Karsai and J. Sztipanovits, “Model-integrated development of cyber-physical systems”, in *Ifip International Workshop on Software Technologies for Embedded and Ubiquitous Systems*, Springer, 2008, pp. 46–54. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-87785-1_5 (visited on Nov. 10, 2016).

- [66] K. Kim, W. Kang, B. Sagong, and H. Seo, “Efficient distributed simulation of hierarchical DEVS models: Transforming model structure into a non-hierarchical one”, in *Simulation Symposium, 2000.(SS 2000) Proceedings. 33rd Annual*, IEEE, 2000, pp. 227–233. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=844920 (visited on Oct. 30, 2016).
- [67] E. Kofman, “Quantized-state control. A method for discrete event control of continuous systems”, *Latin American applied research*, vol. 33, no. 4, pp. 399–406, 2003. [Online]. Available: <http://pdf.easechem.com/pdf/14/v33n4a06.pdf> (visited on Oct. 30, 2016).
- [68] E. Kofman and S. Junco, “Quantized-state systems: A DEVS Approach for continuous system simulation”, *Transactions of The Society for Modeling and Simulation International*, vol. 18, no. 3, pp. 123–132, 2001. [Online]. Available: <http://www.fceia.unr.edu.ar/~kofman/files/qss.pdf> (visited on Oct. 30, 2016).
- [69] E. Kofman, M. Lapadula, and E. Pagliero, “PowerDEVS: A DEVS-based environment for hybrid system modeling and simulation”, *School of Electronic Engineering, Universidad Nacional de Rosario, Tech. Rep. LSD0306*, 2003. [Online]. Available: <http://usuarios.fceia.unr.edu.ar/~kofman/files/lsd0306.pdf> (visited on Oct. 6, 2016).
- [70] Konsortium Project Balanced Manufacturing, *Documentation of BaMa Methodology*, 2015. [Online]. Available: http://bama.ift.tuwien.ac.at/fileadmin/t/bama/Documentation_of_BaMa_Methodology.pdf (visited on Oct. 10, 2016).
- [71] Konsortium Project INFO, “Info - interdisziplinäre forschung zur energieoptimierung in fertigungsbetrieben: Endbericht”, Konsortium Project INFO, Wien, Publizierbarer Endbericht, Sep. 30, 2013, p. 132. [Online]. Available: http://www.projekt-info.org/endbericht/2013-09-30%20publizierbarer_endbericht_final.pdf (visited on Oct. 9, 2016).
- [72] I. Kovacic, K. Orehounig, A. Mahdavi, F. Bleicher, A.-A. Dimitrou, and L. Waltenbereger, “Energy Efficient Production – Interdisciplinary, Systemic Approach through Integrated Simulation”, *Strojarstvo: Journal for Theory and Application in Mechanical Engineering*, vol. 55, no. 1, pp. 17–34, 2013, ISSN: 0562-1887. [Online]. Available: http://publik.tuwien.ac.at/files/PubDat_222974.pdf%20http://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=158086.
- [73] M. Kurttila, M. Pesonen, J. Kangas, and M. Kajanus, “Utilizing the analytic hierarchy process (AHP) in SWOT analysis - a hybrid method and its application to a forest-certification case”, *Forest Policy and Economics*, vol. 1, no. 1, pp. 41–52, May 1, 2000, ISSN: 1389-9341. DOI: 10.1016/S1389-9341(99)00004-0. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389934199000040> (visited on Nov. 6, 2016).

- [74] A. M. Law and S. W. Haider, “Selecting simulation software for manufacturing applications: Practical guidelines & software survey”, *Industrial Engineering*, vol. 21, no. 5, pp. 33–46, 1989. [Online]. Available: <http://dl.acm.org/citation.cfm?id=70078> (visited on Nov. 6, 2016).
- [75] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 2nd ed, ser. McGraw-Hill series in industrial engineering and management science. New York: McGraw-Hill, 1991, 759 pp., ISBN: 978-0-07-036698-5.
- [76] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, “The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 3, pp. 131–145, Jul. 1982, ISSN: 0278-0070. DOI: 10.1109/TCAD.1982.1270004.
- [77] I. Leobner, “Forschungsfabrik zur energieeffizienz: Bama - balanced manufacturing”, Science Brunch 2014 - Leitprojekte für Leitmärkte, Sep. 22, 2014, [Online]. Available: https://www.klimafonds.gv.at/assets/Uploads/Broschren/Science-Brunch-Broschren/2014/KLIEN_2014_ScienceBrunch-Leitprojekte_fuer-Leitmaerkte.pdf (visited on Oct. 30, 2016).
- [78] —, “Modeling of Energy Systems for Complex Simulations”, Dissertation, TU Wien, Wien, 2016. [Online]. Available: http://publik.tuwien.ac.at/files/PubDat_247849.pdf (visited on Oct. 31, 2016).
- [79] I. Leobner, K. Ponweiser, G. Neugschwandtner, and W. Kastner, “Energy efficient production-a holistic modeling approach”, in *Sustainable Technologies (WCST), 2011 World Congress on*, IEEE, 2011, pp. 62–67. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6114239 (visited on Oct. 9, 2016).
- [80] I. Leobner, P. Smolek, B. Heinzl, I. Kovacic, F. Dür, K. Ponweiser, and W. Kastner, “Balanced Manufacturing - a Methodology for Energy Efficient Production Plant Operation”, in *Proceedings of the 10th Conference on Sustainable Development of Energy, Water and Environment Systems (SDEWES 2015)*, Dubrovnik, Croatia, Sep. 27–Oct. 2, 2015.
- [81] G. T. Mackulak, J. K. Cochran, and P. A. Savory, “Ascertaining important features for industrial simulation environments”, *Simulation*, vol. 63, no. 4, pp. 211–221, 1994. [Online]. Available: <http://sim.sagepub.com/content/63/4/211.short> (visited on Nov. 2, 2016).
- [82] A. Mehlhase, “Konzepte für die modellierung und simulation strukturvariabler modelle”, Dissertation, TU Berlin, 2015. [Online]. Available: https://www.depositonce.tu-berlin.de/bitstream/11303/4811/2/mehlhase_alexandra.pdf.

- [83] U. Miekkala and O. Nevanlinna, “Convergence of Dynamic Iteration Methods for Initial Value Problems”, *SIAM Journal on Scientific and Statistical Computing*, vol. 8, no. 4, pp. 459–482, Jul. 1987, ISSN: 0196-5204, 2168-3417. DOI: 10.1137/0908046. [Online]. Available: <http://epubs.siam.org/doi/abs/10.1137/0908046> (visited on Oct. 28, 2016).
- [84] T. Miettinen *et al.*, “Synchronized cooperative simulation: OPC UA based approach”, 2012. [Online]. Available: <https://aaltodoc.aalto.fi/handle/123456789/5201> (visited on Nov. 9, 2016).
- [85] Modelica Association, *Modelica - A Unified Object-Oriented Language for Systems Modeling: Language Specification Version 3.3 Revision 1*, Jul. 11, 2014. [Online]. Available: <https://www.modelica.org/documents/%20ModelicaSpec33Revision1.pdf> (visited on Oct. 8, 2016).
- [86] Modelica Association Project FMI, “Functional Mock-up Interface for Model Exchange and Co-Simulation”, Version 2.0, Jul. 25, 2014. [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf (visited on Oct. 27, 2016).
- [87] Modelisar Consortium, “Functional Mock-up Interface for Co-Simulation”, Version 1.0, Oct. 12, 2010. [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/v1.0/FMI_for_CoSimulation_v1.0.pdf (visited on Oct. 27, 2016).
- [88] R. Mosshammer, F. Kupzog, M. Faschang, and M. Stifter, “Loose coupling architecture for co-simulation of heterogeneous components”, in *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, IEEE, 2013, pp. 7570–7575. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6700394 (visited on Oct. 6, 2016).
- [89] J. Nikoukaran, V. Hlupic, and R. J. Paul, “Criteria for Simulation Software Evaluation”, in *Proceedings of the 30th Conference on Winter Simulation*, ser. WSC ’98, Los Alamitos, CA, USA: IEEE Computer Society Press, 1998, pp. 399–406, ISBN: 978-0-7803-5134-9. [Online]. Available: <http://dl.acm.org/citation.cfm?id=293172.293256> (visited on Nov. 2, 2016).
- [90] J. Nikoukaran, V. Hlupic, and R. J. Paul, “A hierarchical framework for evaluating simulation software”, *Simulation Practice and Theory*, vol. 7, no. 3, pp. 219–231, May 15, 1999, ISSN: 0928-4869. DOI: 10.1016/S0928-4869(98)00028-7. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0928486998000287> (visited on Nov. 2, 2016).
- [91] J. Nikoukaran and R. J. Paul, “Software selection for simulation in manufacturing: A review”, *Simulation Practice and Theory*, vol. 7, no. 1, pp. 1–14, Mar. 15, 1999, ISSN: 0928-4869. DOI: 10.1016/S0928-4869(98)00022-6. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0928486998000226> (visited on Nov. 2, 2016).

- [92] J. Nutaro, P. T. Kuruganti, L. Miller, S. Mullen, and M. Shankar, “Integrated hybrid-simulation of electric power and communications systems”, in *Power Engineering Society General Meeting, 2007. IEEE*, IEEE, 2007, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4275968 (visited on Oct. 5, 2016).
- [93] J. Nutaro, “ADEVS (a discrete EVent system simulator)”, *Arizona Center for Integrative Modeling & Simulation (ACIMS), University of Arizona, Tucson*. Available at <http://www.ece.arizona.edu/nutaro/index.php>, 1999.
- [94] C. Nytsch-Geusen, T. Ernst, A. Nordwig, P. Schneider, P. Schwarz, M. Vetter, C. Wittwer, A. Holm, T. Nouidui, J. Leopold, G. Schmidt, U. Doll, and A. Mattes, “Mosilab: Development of a modelica based generic simulation tool supporting model structural dynamics”, in *4th International Modelica Conference*, Berlin, 2005, pp. 527–535. [Online]. Available: https://www.modelica.org/events/Conference2005/authorindex/online_proceedings/Session6/Session6c3.pdf.
- [95] C. W. Park, J. H. Jeong, and Y. T. Kang, “Energy consumption characteristics of an absorption chiller during the partial load operation”, *International Journal of Refrigeration*, vol. 27, no. 8, pp. 948–954, Dec. 2004, ISSN: 0140-7007. DOI: 10.1016/j.ijrefrig.2004.06.002. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0140700704001057> (visited on Oct. 24, 2016).
- [96] M. Paulus and F. Borggrefe, “The potential of demand-side management in energy-intensive industries for electricity markets in Germany”, *Applied Energy*, The 5th Dubrovnik Conference on Sustainable Development of Energy, Water and Environment Systems, held in Dubrovnik September/October 2009, vol. 88, no. 2, pp. 432–441, Feb. 2011, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2010.03.017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0306261910000814> (visited on Nov. 10, 2016).
- [97] T. Pawletta, C. Deatcu, O. Hagendorf, S. Pawletta, and G. Colquhoun, “DEVS-Based Modeling and Simulation in Scientific and Technical Computing Environments”, in *Devs Integrative MEs Symposium (devs’06)*, vol. 38, 2006, pp. 151–158. [Online]. Available: <https://www.scs.org/confernc/springsim/springsim06/prelimProgram/devs/22.html%20https://www.scs.org/404.html>.
- [98] T. Pawletta, B. Lampe, S. Pawletta, and W. Drewelow, “A DEVS-Based Approach for Modeling and Simulation of Hybrid Variable Structure Systems”, in *Modelling, Analysis, and Design of Hybrid Systems*, S. Engell, G. Frehse, and E. Schnieder, Eds., vol. 279, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 107–129, ISBN: 978-3-540-43812-0. [Online]. Available: http://link.springer.com/10.1007/3-540-45426-8_7 (visited on Oct. 30, 2016).

- [99] A. R. Plummer, “Model-in-the-Loop Testing”, *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 220, no. 3, pp. 183–199, Jan. 1, 2006, ISSN: 0959-6518, 2041-3041. DOI: 10.1243/09596518JSCE207. [Online]. Available: <http://sdj.sagepub.com/lookup/10.1243/09596518JSCE207> (visited on Oct. 28, 2016).
- [100] H. Prähofer, “System Theoretic Formalisms for Combined Discrete-Continuous System Simulation”, *International Journal of General Systems*, vol. 19, no. 3, pp. 219–240, Oct. 1, 1991, ISSN: 0308-1079. DOI: 10.1080/03081079108935175. [Online]. Available: <http://dx.doi.org/10.1080/03081079108935175> (visited on Oct. 30, 2016).
- [101] —, “System Theoretic Foundations for Combined Discrete-Continuous System Simulation”, Dissertation, Johannes Kepler Universität Linz, Linz, Austria, 1992.
- [102] F. Preyser, “An Approach to Develop a User Friendly Way of Implementing DEV&DESS Models in PowerDEVs”, Diploma Thesis, TU Wien, Wien, 2015. [Online]. Available: <http://www.ub.tuwien.ac.at/dipl/2015/AC12315517.pdf> (visited on Oct. 29, 2016).
- [103] F. Preyser, B. Heinzl, P. Raich, and W. Kastner, “Towards Extending the Parallel-DEVs Formalism to Improve Component Modularity”, presented at the ASIM Workshop STS/GMMS 2015, Lippstadt, 2015.
- [104] C. Pühringer, “Using the Modelica language to simulate hybrid models”, TU Wien, Wien, Student Project, Oct. 3, 2016.
- [105] P. Raich, B. Heinzl, F. Preyser, and W. Kastner, “Modeling Techniques for Integrated Simulation of Industrial Systems Based on Hybrid PDEVs”, in *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, Apr. 2016, pp. 1–6, ISBN: 978-1-5090-1158-2. DOI: 10.1109/MSCPES.2016.7480221. [Online]. Available: <http://ieeexplore.ieee.org/document/7480221/?arnumber=7480221%20http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7480221>.
- [106] S. Rohjans, S. Lehnhoff, S. Schütte, S. Scherfke, and S. Hussain, “Mosaik - A modular platform for the evaluation of agent-based Smart Grid control”, in *2013 4th IEEE/PES Innovative Smart Grid Technologies Europe, ISGT Europe 2013*, IEEE, 2013, pp. 1–5, ISBN: 978-1-4799-2984-9. DOI: 10.1109/ISGTEurope.2013.6695486. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6695486.
- [107] T. L. Saaty, “How to make a decision: The analytic hierarchy process”, *European Journal of Operational Research*, Decision making by the analytic hierarchy process: Theory and applications, vol. 48, no. 1, pp. 9–26, Sep. 5, 1990, ISSN: 0377-2217. DOI: 10.1016/0377-2217(90)90057-I. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/037722179090057I> (visited on Nov. 6, 2016).
- [108] —, “The analytical network process”, *Pittsburgh: RWS Publications*, 1996.

- [109] T. L. Saaty and L. G. Vargas, *Decision Making with the Analytic Network Process: ECONOMIC, Political, Social and Technological Applications with Benefits, Opportunities, Costs and Risks*. Springer Science & Business Media, May 14, 2013, 330 pp., ISBN: 978-1-4614-7279-7.
- [110] V. Sanz, A. Urquía, and S. Dormido, “Parallel DEVS and Process-Oriented Modeling in Modelica”, *Proceedings 7th Modelica Conference*, pp. 96–107, Oct. 2009, ISSN: 1650-3740. DOI: 10.3384/ecp09430104. [Online]. Available: <http://www.ep.liu.se/ecp/043/011/ecp09430104.pdf>.
- [111] H. S. Sarjoughian, D. R. Hild, Xiaolin Hu, and R. A. Strini, “Simulation-based SW/HW Architectural Design Configurations for Distributed Mission Training Systems”, *Simulation*, vol. 77, pp. 23–38, 1-2 Jul. 1, 2001, ISSN: 0037-5497. DOI: 10.1177/003754970107700102. [Online]. Available: <http://sim.sagepub.com/cgi/doi/10.1177/003754970107700102> (visited on Nov. 6, 2016).
- [112] H. S. Sarjoughian and B. R. Zeigler, “DEVJSJAVA: Basis for a DEVS-based collaborative M&S environment”, *Simulation Series*, vol. 30, pp. 29–36, 1998. [Online]. Available: https://www.researchgate.net/profile/Hessam_Sarjoughian/publication/243778266_Devsjava_Basis_for_a_devs-based_collaborative_ms_environment/links/5418f6dc0cf203f155adba65.pdf (visited on Oct. 31, 2016).
- [113] R. Schmoll, *Co-Simulation und Solverkopplung: Analyse komplexer multiphysikalischer Systeme*, in collab. with Kassel University Press GmbH, ser. Berichte des Instituts für Mechanik 3/2015. Kassel: Kassel University Press, 2015, 152 pp., ISBN: 978-3-86219-592-3.
- [114] S. Schöps, H. De Gersem, A. Bartel, and M. Clemens, “Dynamic Iteration for Field/Circuit Coupled Problems”, Talk, Karlsruhe Institute of Technology, 2012, [Online]. Available: http://ace2012.math.kit.edu/abstracts/schoeps_degersem_bartel_clemens.pdf (visited on Oct. 27, 2016).
- [115] S. Schütte, S. Scherfke, and M. Sonnenschein, “Mosaik - Smart grid simulation API: Toward a semantic based standard for interchanging smart grid simulations”, *SMARTGREENS 2012 - Proceedings of the 1st International Conference on Smart Grids and Green IT Systems*, no. 2, pp. 14–24, 2012. [Online]. Available: https://mosaik.offis.de/downloads/mosaik_SimAPI_SmartGreens2012.pdf.
- [116] B. Schweizer and D. Lu, “Semi-implicit co-simulation approach for solver coupling”, *Archive of Applied Mechanics*, vol. 84, no. 12, pp. 1739–1769, Dec. 2014, ISSN: 0939-1533, 1432-0681. DOI: 10.1007/s00419-014-0883-5. [Online]. Available: <http://link.springer.com/10.1007/s00419-014-0883-5> (visited on Oct. 28, 2016).
- [117] A. Seila, V. Čerić, and P. Tadikamalla, *Applied Simulation Modeling*. Brooks/Cole-Thomson Learning, 2003. [Online]. Available: <http://bib.irb.hr/prikazirad?rad=129982> (visited on Nov. 6, 2016).

- [118] N. Shariatzadeh, G. Sivard, and D. Chen, “Software Evaluation Criteria for Rapid Factory Layout Planning, Design and Simulation”, *Procedia CIRP*, 45th CIRP Conference on Manufacturing Systems 2012, vol. 3, pp. 299–304, Jan. 1, 2012, ISSN: 2212-8271. DOI: 10.1016/j.procir.2012.07.052. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2212827112002247> (visited on Nov. 2, 2016).
- [119] M. J. Sharma, I. Moon, and H. Bae, “Analytic hierarchy process to assess and optimize distribution network”, *Applied Mathematics and Computation*, vol. 202, no. 1, pp. 256–265, Aug. 1, 2008, ISSN: 0096-3003. DOI: 10.1016/j.amc.2008.02.008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0096300308000957> (visited on Nov. 6, 2016).
- [120] P. Smolek, “Objektorientierte modellierung und dynamische co-simulation mit catia v6 am beispiel von kraftfahrzeugsystemen”, Diploma Thesis, TU Wien, Wien, 2013, 122 pp. [Online]. Available: <http://www.ub.tuwien.ac.at/dipl/2013/AC11200075.pdf> (visited on Oct. 8, 2016).
- [121] H. Song, “Infrastructure for DEVS Modelling and Experimentation”, Diploma Thesis, McGill University, Montreal, Canada, 2006, 149 pp. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.9850&rep=rep1&type=pdf> (visited on Nov. 4, 2016).
- [122] M. Spiryagin, C. Cole, Y. Q. Sun, M. McClanachan, V. Spiryagin, and T. McSweeney, *Design and Simulation of Rail Vehicles*. CRC Press, May 13, 2014, 330 pp., ISBN: 978-1-4665-7567-7.
- [123] M. Striebel, “Multirate - Introduction”, Talk, TU Wien, May 31, 2011.
- [124] Tag Gon Kim, Chang Ho Sung, S.-Y. Hong, Jeong Hee Hong, Chang Beom Choi, Jeong Hoon Kim, Kyung Min Seo, and Jang Won Bae, “DEVSIM++ Toolset for Defense Modeling and Simulation and Interoperation”, *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, vol. 8, no. 3, pp. 129–142, Jul. 1, 2011, ISSN: 1548-5129, 1557-380X. DOI: 10.1177/1548512910389203. [Online]. Available: <http://dms.sagepub.com/cgi/doi/10.1177/1548512910389203> (visited on Oct. 31, 2016).
- [125] T. W. Tewoldeberhan, A. Verbraeck, E. Valentin, and G. Bardonnnet, “Software Evaluation and Selection: An Evaluation and Selection Methodology for Discrete-event Simulation Software”, in *Proceedings of the 34th Conference on Winter Simulation: EXPLORING NEW FRONTIERS*, ser. WSC ’02, San Diego, California: Winter Simulation Conference, 2002, pp. 67–75, ISBN: 978-0-7803-7615-1. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1030453.1030465> (visited on Nov. 4, 2016).
- [126] The MathWorks, *MATLAB Documentation*, version 9.1 (R2016b), Natick, Massachusetts, Oct. 8, 2016. [Online]. Available: <http://www.mathworks.com/help/matlab/index.html>.

- [127] ———, *Simscape Documentation*, version 4.1 (R2016b), Natick, Massachusetts, Oct. 8, 2016. [Online]. Available: <https://www.mathworks.com/help/physmod/simscape/>.
- [128] S. Thiede, *Energy Efficiency in Manufacturing Systems*, ser. Sustainable Production, Life Cycle Engineering and Management. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN: 978-3-642-25913-5 978-3-642-25914-2. [Online]. Available: <http://link.springer.com/10.1007/978-3-642-25914-2> (visited on Oct. 3, 2016).
- [129] M. Trecka, “Co-simulation for performance prediction of innovative integrated mechanical energy systems in buildings”, Dissertation, Technische Universiteit Eindhoven, 2008. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.472.4041&rep=rep1&type=pdf> (visited on Oct. 25, 2016).
- [130] U.S. Department of Energy, *EnergyPlus Version 8.6 Documentation: Getting Started*, version 8.6, Sep. 30, 2016. [Online]. Available: https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v8.6.0/GettingStarted.pdf (visited on Oct. 8, 2016).
- [131] O. Vaculín, W. R. Krüger, and M. Valášek, “Overview of Coupling of Multibody and Control Engineering Tools”, *Vehicle System Dynamics*, vol. 41, no. 5, pp. 415–429, May 1, 2004, ISSN: 0042-3114. DOI: 10.1080/00423110412331300363. [Online]. Available: <http://dx.doi.org/10.1080/00423110412331300363> (visited on Oct. 29, 2016).
- [132] A. J. Van Der Schaft and J. M. Schumacher, *An Introduction to Hybrid Dynamical Systems*, ser. Lecture notes in control and information sciences. London ; New York: Springer, 2000, vol. 251, 174 pp., ISBN: 1-85233-233-6.
- [133] J. M. Vaterlaus and B. J. Higginbotham, “Qualitative program evaluation methods”, in *The Forum for Family and Consumer Issues*, vol. 16, 2011.
- [134] R. Verma, A. Gupta, and K. Singh, “A Critical Evaluation and Comparison of Four Manufacturing Simulation Softwares”, *Kathmandu University Journal of Science, Engineering and Technology*, vol. 5, no. 1, pp. 104–120, 2009, ISSN: 1816-8752. DOI: 10.3126/kuset.v5i1.2851. [Online]. Available: <http://www.nepjol.info/index.php/KUSET/article/view/2851> (visited on Nov. 4, 2016).
- [135] L. Völker, *Untersuchung des Kommunikationsintervalls bei der gekoppelten Simulation*, ser. Karlsruher Schriftenreihe Fahrzeugsystemtechnik Bd. 6. Karlsruhe: KIT Scientific Publ, 2011, 174 pp., ISBN: 978-3-86644-611-3.
- [136] G. Wainer, “CD++: A toolkit to develop DEVS models”, *Software: Practice and Experience*, vol. 32, no. 13, pp. 1261–1306, 2002. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/spe.482/abstract> (visited on Oct. 31, 2016).

- [137] Y.-H. Wang, “Discrete event simulation on a massively parallel computer.”, Dissertation, University of Arizona, 1992. [Online]. Available: <http://arizona.openrepository.com/arizona/handle/10150/185913> (visited on Oct. 30, 2016).
- [138] Y.-H. Wang and B. P. Zeigler, “Extending the DEVS formalism for massively parallel simulation”, *Discrete Event Dynamic Systems: Theory and Applications*, vol. 3, pp. 193–218, 2-3 Jul. 1993, ISSN: 0924-6703, 1573-7594. DOI: 10.1007/BF01439849. [Online]. Available: <http://link.springer.com/10.1007/BF01439849> (visited on Oct. 30, 2016).
- [139] M. Wetter, “Co-simulation of building energy and control systems with the building controls virtual test bed”, *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185–203, Sep. 2011, ISSN: 1940-1493, 1940-1507. DOI: 10.1080/19401493.2010.518631. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/19401493.2010.518631> (visited on Sep. 27, 2016).
- [140] M. Wetter and T. Noudui, *Building Controls Virtual Test Bed - User Manual Version 1.6.0*, version 1.6.0, Berkeley, CA, Apr. 20, 2016. [Online]. Available: <http://simulationresearch.lbl.gov/bcvtb/releases/1.6.0/doc/manual/bcvtb-manual.pdf>.
- [141] J. Zehetner, G. Stettinger, H. Kokal, and B. Toye, “Echtzeit-co-simulation für die regelung eines motorprüfstands”, *ATZ - Automobiltechnische Zeitschrift*, vol. 116, no. 2, pp. 40–45, Feb. 2014, ISSN: 0001-2785, 2192-8800. DOI: 10.1007/s35148-014-0042-x. [Online]. Available: <http://link.springer.com/10.1007/s35148-014-0042-x> (visited on Oct. 28, 2016).
- [142] B. P. Zeigler, H. Prähofer, and T. G. Kim, *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2000, 536 pp., ISBN: 978-0-12-778455-7.
- [143] Z. Zhai, “Developing an integrated building design tool by coupling building energy simulation and computational fluid dynamics programs”, Dissertation, Massachusetts Institute of Technology, 2003. [Online]. Available: <http://dspace.mit.edu/handle/1721.1/17617> (visited on Oct. 28, 2016).
- [144] W. Zhu, S. Pekarek, J. Jatskevich, O. Wasynczuk, and D. Delisle, “A Model-in-the-Loop Interface to Emulate Source Dynamics in a Zonal DC Distribution System”, *IEEE Transactions on Power Electronics*, vol. 20, no. 2, pp. 438–445, Mar. 2005, ISSN: 0885-8993. DOI: 10.1109/TPEL.2004.842973. [Online]. Available: <http://ieeexplore.ieee.org/document/1408008/> (visited on Oct. 28, 2016).