TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Methods for Hybrid Modeling and Simulation-Based Optimization in Energy-Aware Production Planning

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften (Dr.techn.)
unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker
Institut für Analysis und Scientific Computing
E101

eingereicht an der Technischen Universität Wien
Fakultät für Mathematik und Geoinformation

von

Dipl.-Ing. Dipl.-Ing. Bernhard Heinzl, BSc BSc
Matrikelnummer: 00626050
Oberpeilstein 1
A-4153 Peilstein

Wien, am 14. Mai 2020

Abstract

Production Planning (PPC) optimization in the operation of industrial production plants enables to achieve energy savings and improve energy efficiency. To provide decision support in production planning, simulation-based methods are particularly well-suited for complex production system to assess different planning scenarios and evaluate them with regard to their energy demand.

To this end, a hybrid discrete/continuous modeling approach based on the Discrete-Event System Specification (DEVS) is investigated in this thesis. Hybrid modeling allows to accurately capture material flows as discrete entities on one hand and continuous energy flows on the other hand, in the production as well as for other domains, such as heating or cooling, up to the thermal building envelope, including their dynamic interactions.

A meta-heuristic optimization procedure for operative production planning evaluates energy efficiency together with other production goals (delivery tardiness, storage costs, etc.) as part of a multi-objective optimization. The procedure follows a hybrid Variable Neighborhood Search (VNS) meta-heuristic and uses the hybrid simulation for the energetic evaluation of production programs. The VNS uses special energy-related operators, for example to optimize the set-up times of heating ovens, in order to save energy.

Furthermore, to support the modeling process of such DEVS-based simulations in practical applications, a domain-specific model abstraction is designed, which allows to describe hybrid simulation models in an intuitive way. The method is based on a Model-Driven Engineering (MDE) process and defines a component-based meta-model for production systems. Concrete implementations can be derived from the abstract specification by means of model transformations.

Proof-of-concept implementations are realized to evaluate the individual contributions and case studies from industry demonstrate the practical applicability. The hybrid simulation approach was implemented in its entirety as a simulator prototype, together with reusable model components, which can be used by practitioners to design new application models with reduced efforts. Altogether, these methods contribute to the integration of energy efficiency aspects into modern PPC systems and to increase energy efficiency in production.

Kurzfassung

Im operativen Betrieb von industriellen Anlagen lassen sich durch optimierte Produktionsplanung und -steuerung (PPS) gezielt Energieeinsparungen erreichen und die Energieeffizienz verbessern. Zur Entscheidungsunterstützung bieten sich gerade bei komplexen Produktionssystemen simulationsbasierte Methoden an, die es ermöglichen, unterschiedliche Planungsszenarien zu vergleichen und hinsichtlich ihres Energieeinsatzes zu bewerten.

Dafür wird in der vorliegenden Arbeit ein hybrider diskret/kontinuierlicher Modellierungsansatz untersucht, der auf der Discrete-Event System Specification (DEVS) basiert. Die hybride Modellierung erlaubt es einerseits die Materialflüsse als diskrete Entitäten, andererseits auch die zeitkontinuierlichen Energieflüsse sowohl in der Produktion als auch für andere Domänen, wie Heizung oder Kühlung, bis hin zur thermischen Gebäudehülle akkurat zu erfassen, mitsamt ihrer dynamischen Interaktionen.

Ein meta-heuristisches Optimierungsverfahren zur operativen Produktionsplanung bewertet die Zielgröße der Energieeffizienz gemeinsam mit anderen Produktionszielen (Liefertreue, Lagerhaltungskosten, etc.) im Rahmen eines multikriteriellen Zielsystems. Das Verfahren basiert auf einer Variable Neighborhood Search (VNS) und verwendet die hybride Simulation zur energetischen Bewertung der Produktionsprogramme. Die VNS verwendet spezielle energiebezogene Operatoren, beispielsweise zur Optimierung der Rüstzeiten von Heizöfen, um unnötigen Energieverbrauch einzusparen.

Zur gezielten Unterstützung des Modellierungsprozesses DEVS-basierter Simulationen in der praktischen Anwendung wird außerdem eine domänenspezifische Modellabstraktion konzipiert, die es erlaubt, derartige hybride Simulationsmodelle auf intuitive Art und Weise zu beschreiben. Die Methode basiert auf einem Model-Driven Engineering (MDE) Prozess und definiert ein komponentenbasiertes Meta-Modell für Produktionssysteme. Mittels Modelltransformationen lassen sich aus der abstrakten Spezifikation konkrete Implementierungen ableiten.

Zur Evaluierung der einzelnen Beiträge werden prototypische Implementierungen entwickelt. Fallstudien aus der Industrie dienen zur Demonstration der praktischen Anwendbarkeit. Der hybride Simulationsansatz wurde in seiner Gesamtheit als prototypischer Simulator implementiert, zusammen mit wiederverwendbaren Modellkomponenten, mit deren Hilfe mit reduziertem Aufwand neue Applikationsmodelle entworfen werden können. Insgesamt tragen diese Methoden dazu bei, Energieeffizienzaspekte in moderne PPS-Systeme zu integrieren und damit die Energieeffizienz in der Produktion zu steigern.

Contents

Abstract i

Kurzfassung iii

Contents v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Objective and Hypotheses . . . 3
- 1.3 Scope of the Work . . . 4
- 1.4 Methodology . . . 6
- 1.5 Structure of the Thesis . . . 7

2 Background 9
- 2.1 Energy Efficiency in Industry . . . 9
  - 2.1.1 Energy in Industry . . . 9
  - 2.1.2 Energy Efficiency . . . 11
  - 2.1.3 Improving Energy Efficiency in Industry . . . 12
- 2.2 Production Planning and Control Systems . . . 14
- 2.3 Industrial Information Systems . . . 16

3 Related Work 19
- 3.1 Literature Review on Simulation-based Optimization in Production . . . 19
  - 3.1.1 Optimization Methods . . . 19
  - 3.1.2 Meta-heuristics . . . 20
  - 3.1.3 Optimization of Energy Efficiency in Production . . . 21
  - 3.1.4 Simulation-based Methods for Energy Efficiency in Production 22
  - 3.1.5 Discourse . . . 24
- 3.2 Coupled Simulation . . . 25
  - 3.2.1 Methods for Simulation Coupling . . . 25
  - 3.2.2 Hybrid Co-Simulation . . . 28
  - 3.2.3 Challenges . . . 30
- 3.3 Co-Simulation for Energy Efficiency . . . 32
  - 3.3.1 Reference Model . . . 32

3.3.2 Co-simulation Implementation . . . 33
3.3.3 Results . . . 35
3.3.4 Discourse . . . 36

4 Concept for Simulation-Based Energy-Aware PPC 39
4.1 Integrating Energy into PPC . . . 39
4.2 Conceptual Modeling for Simulation . . . 41
4.2.1 Frameworks, Tools and Languages for Conceptual Modeling . . . 41
4.2.2 The Model Continuity Problem . . . 42
4.2.3 Component-based Modeling . . . 42
4.3 The Cube Concept . . . 43
4.4 Engineering Workflow . . . 45

5 Hybrid Simulation of Production Systems 49
5.1 Introduction . . . 49
5.2 The hyPDEVS Formalism . . . 50
5.2.1 hyPDEVS Atomic . . . 51
5.2.2 hyPDEVS Coupled . . . 53
5.2.3 Legitimacy . . . 55
5.2.4 Closure under Coupling . . . 56
5.2.5 Simulation Execution . . . 56
5.2.6 Handling ODEs . . . 58
5.2.7 Remarks . . . 60
5.3 Cube Modeling . . . 60
5.3.1 Interfaces . . . 61
5.3.2 Entity Modeling . . . 62
5.4 Cube Library . . . 63
5.5 Example: Conveyor Oven . . . 63
5.5.1 Cube Model . . . 63
5.5.2 Remarks . . . 67
5.5.3 hyPDEVS Model . . . 68
5.5.4 Remarks . . . 71
5.6 Implementation . . . 74
5.6.1 First Prototype . . . 74
5.6.2 Stand-alone Simulator . . . 74
5.7 Case Study 1: Simple Production Line . . . 75
5.7.1 Model Description . . . 75
5.7.2 Scenario 1 . . . 77
5.7.3 Scenario 2 . . . 80
5.7.4 Validation . . . 82
5.8 Case Study 2: Advanced Production Line . . . 83
5.8.1 Model Description . . . 83
5.8.2 Validation . . . 86
5.9 Discourse . . . 86

6 Multi-objective Production Planning Optimization **89**
6.1 Introduction . . . 89
6.2 Simulation-based Optimization Strategy . . . 90
6.2.1 Meta-heuristics . . . 90
6.2.2 Simulation-Optimization Cycle . . . 91
6.3 Problem Description . . . 92
6.3.1 Multi-objective Problem Formulation . . . 93
6.3.2 System under Study . . . 94
6.3.3 Decision Variables . . . 95
6.4 GVNS Optimization Method . . . 96
6.4.1 Initial Solution . . . 97
6.4.2 Generalized Cost Function . . . 98
6.4.3 Neighborhood Structures . . . 100
6.4.4 Variable Neighborhood Descent . . . 101
6.4.5 Acceptance Criterion . . . 102
6.5 Case Study Experiments . . . 102
6.5.1 Implementation . . . 102
6.5.2 Scenario 1: One Week Planning Horizon . . . 103
6.5.3 Scenario 2: Realistic Energy Price . . . 108
6.6 Discourse . . . 114

7 Framework for Model Engineering **117**
7.1 Introduction . . . 117
7.2 Model-driven Engineering Methodology . . . 118
7.2.1 Overview . . . 119
7.2.2 Modeling Languages . . . 120
7.2.3 Meta-Modeling . . . 121
7.2.4 Model Transformations . . . 122
7.2.5 Model-Driven Architecture . . . 123
7.2.6 Model-Driven vs. Model-Based Terminology . . . 124
7.3 Modeling Framework for Cube-Based Application Engineering . . . 125
7.3.1 hyPIM Metamodel . . . 127
7.3.2 hyPDEVS Metamodel . . . 128
7.3.3 Model Transformations . . . 129
7.3.4 Cube Library . . . 129
7.4 Implementation . . . 131
7.4.1 Model to Model Transformation . . . 132
7.4.2 Code Generation . . . 132
7.5 Discourse . . . 132

8 Conclusion **135**
8.1 Discussion . . . 135
8.2 Future Work . . . 138

A Oven Cube hyPDEVS Implementation **141**

List of Figures **147**

List of Tables **151**

List of Algorithms **153**

Glossary **155**

Bibliography **161**

CHAPTER 1

Introduction

1.1 Motivation

Energy efficiency has become an increasingly important topic in industrial engineering in recent years, on the one hand due to legislative pressure, but also because of the significant potential for economic savings as well as reducing ecological impact [184, 59]. It is estimated that the manufacturing industry, as one of the largest energy consumers, has up to 35-60% energy savings potential, depending on the sector [33]. Moreover, 97% of production companies consider the topic of energy efficiency as important, according to a survey [190]. In addition to the primary goals of factory planning, inventory and capacity optimization, the planning and reduction of energy consumption will become an increasingly important factor.

Besides designing energy-efficient production plants, energy can be saved by improving the operational performance of existing plants, primarily through efficient Production Planning and Control (PPC). Energy-aware PPC strategies can influence energy demand and energy costs during operation, for example by shifting the production of energy-intensive products to the night hours where energy is often cheaper, or during midday in case more solar energy is available [129]. However, the PPC systems currently used in industry for the most part do not consider energy aspects at all, or if they do then only in a very simplified way by means of static calculations [261].

The transition towards energy-aware PPC therefore requires to provide the planning operator with sophisticated decision-support tools that are able to represent and analyze the complexity of the planning situation [249]. However, these tools cannot evaluate the energy consumption of production as an isolated target, but instead have to weigh it against other production goals, such as delivery tardiness, storage costs or throughput. Being able to view the problem as a whole requires complex multi-criteria optimization.

Energy efficiency must be seen as part of a multi-objective system of production targets. Such complex multi-objective optimization problems with often time-dependent constraints are hard to solve for real-world industrial production planning problems. Modern solutions typically rely on heuristic or meta-heuristic methods [295]. Meta-heuristics allow to explore the search space efficiently and effectively, especially if they are customized to the particular problem [129]. These methods evaluate different planning scenarios in a systematic and iterative manner by quantifying their fitness for a given optimization target [115] in order to ultimately achieve an optimal configuration.

As one of the challenges, these methods need to be able to reliably and accurately assess the impact of production on the overall energy demand. Because of the complexity of modern production systems, static calculation methods are no longer sufficient or would be difficult to manage. Instead, simulation-based methods are gaining interest because they enable to capture the complexity and dynamics of real-world problems without the limiting assumptions many other approaches have. Simulations can deliver calculated energy demands from production schedule forecasts and thus support decision-making in production planning [280, 152, 137, 138]. They allow to consider more complex systems than conventional analytical models, while offering more accurate predictions and overall improving planning quality [181, 252].

In order to be able to perform comprehensive energetic investigations within industrial facilities, the underlying simulation models have to capture the complexity of the underlying system with sufficient level of detail. The models have to incorporate aspects from different engineering domains, in particular production machinery with its material flow, energy infrastructure, logistics and building physics. For example, the interaction of a production oven generating waste heat affects heating and cooling demand of the surrounding building. Similarly, the actual setup time for pre-heating the oven depends on such conditions as the temperature of the products having been produced before, and this setup time in turn affects the production throughput and scheduling [129]. Incorporating energy considerations in production logistics simulations with their time-dependent interactions in an accurate manner requires advanced modeling and simulation approaches that combine discrete and continuous dynamics. While material flow entities can intuitively be modeled using discrete-event methods, energy flow (including transient effects) is best described using time-continuous dynamics with differential equations. Integrating discrete and continuous modeling methods as part of a *hybrid* modeling and simulation approach remains a challenging task [39]. As [235] points out, only few publications so far focus on hybrid systems in the context of production simulation.

Hybrid discrete/continuous simulation in practice requires coupling discrete-event methods with differential equation solvers in a way that is not only efficient (in terms of runtime) but also formally sound (in order to produce accurate results). One common approach is to couple discrete and continuous simulation tools as part of a so-called co-simulation [130]. As a drawback of that approach, the user is forced to split the overall model into different sub-models along the boundary of discrete/continuous modeling, thereby loosing component modularity. It quickly becomes cumbersome to maintain and reuse these

kinds of models. A more promising approach in this context uses a formal model description based on an extended Discrete-Event System Specification (DEVS) for hybrid systems [320]. Discrete-Event System Specification (DEVS) has a sound basis as a formal model description in the academic field. In practical industrial applications, however, it has experienced little adoption so far [130, 135]. A comparison with other co-simulations also shows the potential benefits of the DEVS approach, especially for modular component-based and hierarchical hybrid modeling as well as tighter integration of continuous and discrete model parts while maintaining potentially better performance [135]. So far, no alternative hybrid discrete/continuous simulation approaches are known that aim at industrial planning optimization [261].

Practical computer-aided decision support solutions for deployment in real-world applications require a lot of manual development effort to derive simulation models and optimization algorithms of a particular system [172]. In addition, engineers need to overcome the gap that exists between solutions proposed by research and the actual needs and implementations in the industry. In [47], the authors investigate this gap for integrating energy efficiency into production management and they conclude that many of the existing tools are not suitable for practical deployment due to the lack of practicability and comprehensiveness. Modern solutions based on the Model-Driven Engineering (MDE) methodology can help to mitigate this gap and systematically manage the development process of concrete application models [40].

Managing the system complexity is a key element for real-world applicability. In software design, this can be enabled mainly by two approaches: abstraction and separation of concerns [194]. Abstraction can be achieved through conceptual modeling, which avoids unnecessary implementation details during the early development stages and helps communicating concrete solutions to production managers and other stakeholders [172]. Separation of concerns can be achieved by by employing component-based modeling that allows to compose models from reusable components in a bottom-up manner. Model components provide modularity in simulation, which is a necessary prerequisite for model reuse in an attempt to reduce the effort necessary to develop new application models.

1.2 Research Objective and Hypotheses

To overcome the obstacles mentioned above, this thesis conducts research into methods for simulation-based optimization in production planning of complex production systems. The overarching aim of this work is to support the integration of energy aspects into modern PPC solutions. In particular, we ask the question:

> *How can we facilitate the use and adoption of simulation-based optimization techniques for decision support in practical energy-aware production planning to increase energy efficiency in industry?*

This is not a trivial task for real-world applications as meta-heuristic optimization techniques have to be employed that require hybrid simulation in the background for assessing energy consumption in conjunction with the material flow. In addition, methods are required that support the engineering of such models in practice and to simplify modeling for application engineers.

From this research question, we can derive three research hypotheses that are to be examined in the scope of this thesis:

Hypothesis 1: *A simulation approach based on DEVS is feasible for developing hybrid simulation models of real-world production systems that are modular, reusable and facilitate separation of concerns.*

Hypothesis 2: *The integration of material and energy flow simulation enables meta-heuristic optimization techniques to provide energy-aware production planning.*

Hypothesis 3: *A Model-Driven Engineering (MDE) workflow provides a suitable methodology for engineering hybrid simulation models in practical applications. It offers an intuitive model abstraction and formally sound way of model specification that enables modular composition, reuse and allows to derive and configure hybrid DEVS-based simulations.*

We are concerned in particular with practical requirements, such as modularity, component reuse and runtime efficiency, that are necessary to apply the methods in practice.

As the DEVS-based approach to hybrid simulation is not yet established in industrial applications, we are especially interested in how it performs against common solutions using co-simulation. Intuitively, it seems that DEVS-based simulation entails more development effort to implement a working simulation. However, it promises some interesting advantages that are to become apparent only after conducting some real-world case studies.

1.3 Scope of the Work

A central aspect is the practical applicability of the developed methods. For this purpose, it is important to support the implementation process on one hand and to put special focus on a high reusability of the simulation models on the other hand. To this end, novel contributions in three areas are presented:

Hybrid Simulation: A method for hybrid discrete/continuous simulation of production systems is investigated that employs a formal model description based on DEVS. The hybrid approach allows to accurately capture the material flows (discrete) in a production plant as well as the energy flows (continuous, using differential equations) including their dynamic interactions. Unlike co-simulation, which often uses ad hoc coupling and is

very error-prone in implementation, the DEVS-based method is component-oriented and allows integrated hybrid modeling at component level, thus allowing for component reuse. The practical applicability is demonstrated in several case studies that are based on real applications, for which a library of reusable hybrid and modular component models is developed. In a first step, an initial prototype implementation is conducted in MATLAB, which then serves as a basis to develop a standalone hybrid simulator implementation.

Meta-heuristic multi-objective Optimization: Based on the hybrid simulation, a multi-criteria optimization method for operative production planning is developed, which specifically considers the energy consumption in the target system. The optimization method aims at sequencing and time scheduling a given list of production jobs while minimizing energy demand together with other production goals. This includes optimizing setup times to reduce energy demand and increase production throughput. The procedure combines a Variable Neighborhood Search (VNS) meta-heuristic with Variable Neighborhood Descent (VND) and Simulated Annealing (SA) for local search and diversification, and employs the simulation for energetic evaluation of different production schedules. The Variable Neighborhood Search (VNS) uses energy-related operators and neighborhoods, for example to optimize the setup times of thermal processes in order to save unnecessary energy consumption. The combination of General Variable Neighborhood Search (GVNS) meta-heuristic with hybrid simulation for industrial production planning is novel in the literature.

Model and Simulation Engineering: In an effort to support the engineering process of concrete production planning applications, and especially the modeling process of domain-specific DEVS-based simulations, an approach for conceptual modeling of interdisciplinary production systems is developed that allows the user to describe such models by instantiating, connecting and configuring component models in an intuitive manner by abstracting away irrelevant implementation details from the user. The method is based on a model-driven development process and defines a domain-specific specialization in the form of a component-based meta-model for the domain of production systems. From the abstract model description, system-specific implementations of executable simulations can be derived by means of model transformations. This separation of high-level specification and specific implementation promises better flexibility in the modeling process and significant advantages in managing the complexity of real-world applications.

It is important to note that we deliberately focus on the methods for applying hybrid simulation-based optimization in industry rather than on the application cases themselves. For one thing, the concrete parametrization of the case study simulations is not in the scope of this work. There are several companion publications that have resulted from the same research projects, in which the current thesis is also located, that explain these complementary aspects in more detail, in particular [256, 261]. However, the fact that the case studies are still derived from real production plants ensures the applicability of the methods and relevance for real-life applications.

1.4 Methodology

In this section, the basic methodology is outlined for testing the stated hypotheses. As mentioned, we are concerned more with practical requirements than a pure theoretical analysis, which is why we base our work on a research methodology from applied research. The work follows an experimental design science methodology for information systems research [305, 214] supported by use cases to frame the research activities. The use cases stem from multiple research projects and are derived from real production facilities. This ensures that the research stays goal-oriented and that the results are relevant to practice and applicable. In particular, use cases are considered from an industrial bakery.

These use cases impose certain requirements on the end result regarding scope and functionality as well as performance and usability. In addition, special focus lies on reusability and modularity of hybrid model components.

In a first step, a survey of the current state of the art is conducted to identify relevant related work on simulation-based methods for energy efficiency in industry. Special focus is being put on hybrid discrete/continuous simulation, and we go into detail regarding hybrid co-simulation. We discuss some of the challenges that face these current approaches when applying them in practice and then outline a general concept how hybrid simulation-based optimization can be integrated into a planning module as part of an industrial information system.

As part of the design and development phase, a DEVS-based modeling formalism is investigated regarding its applicability for hybrid simulation in an industrial context. Modeling and execution issues are highlighted and test cases are implemented in the experimental MatlabDEVS simulator. This also serves as a basis for a first use case implementation that allows to estimate how this approach is suited for real-world application. Common white-box modeling techniques are used for deriving the simulation component models. Based on the first proof of concept, a standalone hybrid DEVS simulator is developed that serves for execution of the use case examples. These artifacts are iteratively refined and extended during development.

In the next step, we investigate the benefits of using hybrid simulation within a simulation-based meta-heuristic optimization method and show how optimization results can be improved using more accurate modeling. We demonstrate the feasibility of this method on a flow shop scheduling problem of an industrial bakery. Different scenarios are compared and we highlight the potential benefit of considering energy as an optimization target.

Subsequently, a meta-model is formalized that captures the domain-specific features and concepts for high-level (platform-independent) model specification, following a Model-Driven Engineering (MDE) paradigm with iterative and incremental refinement. In addition, model-to-model and model-to-text transformations are developed that are able to generate platform-specific executable implementations from the high-level specification. Besides a MatlabDEVS implementation, we are also interested in transformations into purely discrete implementations. By implementing the meta-model and model

transformations, we can provide proof-of-concept implementations of a case study that demonstrates the modeling workflow.

Finally, an evaluation of the modeling workflow as well as the generated simulation models by means of feasibility analysis in case studies concludes whether the approach is suitable for application in an industrial context. In particular, we highlight the balance between flexibility and complexity of the engineering method and whether or not the envisioned modeling workflow is able to improve reuse of hybrid component-based models.

1.5 Structure of the Thesis

The thesis is organized as follows: In Chapter 2, we present some relevant background regarding energy efficiency in industry and PPC systems. Then in Chapter 3, we review the related work and relevant state of the art published in the literature. Based on this, we outline the general concept for energy-aware production planning within industrial information systems in Chapter 4. We also discuss an approach for conceptual modeling based on components, called Cubes, and how this approach frames the subsequent research. In Chapter 5 we formalize hybrid DEVS-based simulation, develop a library of Cube components and demonstrate its application on different case studies. Chapter 6 then goes into detail regarding meta-heuristic simulation-based optimization for production planning. It presents a novel procedure based on a hybrid VNS/SA algorithm and applies it on a case study in different scenarios. Then in Chapter 7, we discuss a formalization of the model and simulation engineering workflow based on a MDE methodology. We conclude in Chapter 8 with a discussion of the proposed research hypotheses and outline possible future work.

Parts of this thesis have also been published in other papers over the course of multiple research projects. The most relevant ones are [136, 135, 229, 130, 252, 133, 132, 129, 28, 29, 134].

CHAPTER 2

Background

Due to the application-oriented nature of our work, we first present some relevant background for the industrial domain. We introduce some relevant terminology and discuss Production Planning and Control (PPC) systems as well as industrial information systems.

2.1 Energy Efficiency in Industry

2.1.1 Energy in Industry

In order to be able to manufacture products in an industrial plant, it is necessary to supply the plant with physical energy, in one form or another. This is called *energy utilization* or *energy consumption*. From a purely physical standpoint, the latter term is obviously not correct considering that energy cannot be 'consumed' but only converted to other forms of energy. From a more practical standpoint, however, it could be argued that, since usable energy is converted during production into energy that is no longer usable (or used) in a technical process (e.g. diffuse waste heat), the energy can be regarded as "consumed" in this sense[1].

Before arriving at the consumer, the energy is usually converted in a multistage process, starting from *primary energy* that occurs in nature (e.g. oil, but also wind or solar energy), via *secondary energy* for transport and storage (e.g. gasoline or electric energy), to *final energy* (e.g. electric energy after transport losses), as depicted in Figure 2.1. The final energy is the one being billed to the customer [140]. Within the plant, the final energy is further converted into *usable energy* for production (e.g. mechanical energy for machining or heat from electricity). All these conversion processes come with energy

[1] This notion was later picked up within the research project Balanced Manufacturing (BaMa), to guide the development of a method for attributing CO_2 footprint to processes and products within the production. See [256] for more details.

losses – in many cases less than 30% of the primary energy arrive as usable energy. However, the actual efficiency heavily depends on the conversions involved [140]. During production, the majority of usable energy dissipates as waste heat. Experimental studies on profiling the power consumption during machining showed that for metal cutting operations (milling, turning, etc.), only between 3% and 30% of the energy that is used to power electric drives, spindles etc., ends up stored in the material itself (e.g. as residual stresses, chip formation) [172, 82].

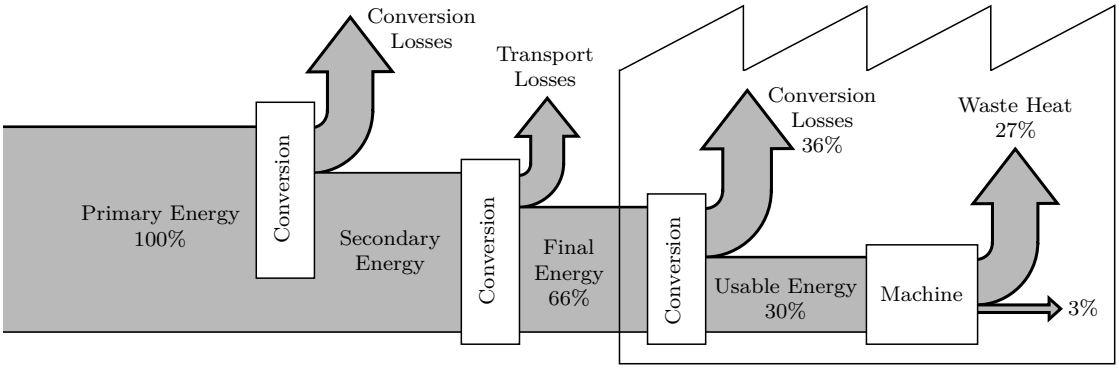


Figure 2.1: Energy conversion chain in industry

When looking at Figure 2.1, one interesting conclusion emerges: An increase in energy savings is all the more effective (in terms of primary energy, e.g. for climate protection), the later it is achieved in the energy conversion chain. For example, saving 1 kWh of compressed air in production can lead to saving 20 kWh of primary energy in the form of coal [140]. This demonstrates the importance of investigating energy efficiency in production for its (magnified) ecological impact.

For the economic success of industrial companies, however, energy consumption itself is often not as important as the associated energy consumption *costs*. In practice, companies will not necessarily optimize energy alone, but instead together with other (economic) targets, as part of a holistic optimization within an energy-aware PPC solution [261]. Hereby, the costs represent a common economic reference unit for different production factors (e.g. storage, transport), which allows to make these different factors comparable in terms of their economic impact (e.g. storage costs compared against energy costs).

An important factor in defining energy consumption costs are the continuous changes in energy *prices* [250]. This is especially true for large companies, which often buy electric energy on the electricity exchange market [261]. The fact that electrical energy is difficult to store makes it necessary to constantly adjust the production and consumption. The price of electricity may thus change over the course of the day according to supply and demand [202]. At night, when there is less demand, electricity is usually cheaper.

For the consumer, a variable energy price implies that saving energy does not necessarily mean the same as saving energy costs. Instead, the variability in energy prices opens an additional degree of freedom: Even if the overall energy demand stays the same, it might

still be possible to save energy costs, e.g. by shifting the production into the nighttime when energy costs are lower [261]. However, this approach almost certainly affects other aspects (e.g. changing storage costs or personnel demand[2]) and it must be assured to not negatively affect the production goals (e.g. delivery timeliness). The energy (and energy costs) must therefore always be considered together with other production parameters.

Other possibilities to save energy costs (without necessarily saving the amount of energy consumed) include production smoothing, i.e. leveling out peak energy loads that would cause higher energy prices. Production smoothing is often done as part of longer-term production planning, which covers a longer time span than short-term PPC as well as strategic decisions [154]. More details on PPC are discussed in Section 2.2.

2.1.2 Energy Efficiency

In general terms, technical *energy efficiency* commonly measures the ratio of usable energy output to energy input into a process [211]:

$$\eta_{\text{EnEff}} = \frac{E_{\text{usable}}}{E_{\text{in}}}. \tag{2.1}$$

Regarding the energy input E_{in}, a distinction can be made between different levels of energy efficiency, depending on whether primary, secondary or final energy is used. The more challenging part, however, is determining what constitutes a usable energy output E_{usable}. This strongly depends on the context of investigation [186]. In [211], the author defines energy efficiency even more broadly as

$$\eta_{\text{EnEff}} = \frac{\text{useful output}}{E_{\text{in}}} \tag{2.2}$$

by potentially considering any form of *useful output* of a production process, system or economic sector, giving rise to a number of different categories of energy efficiency indicators (physical-thermodynamic, economic, etc.). Useful output could for example be the number of pieces produced or the economic turnover generated from them. Since we are more concerned with the technical perspective on energy efficiency rather than the economic one, we will focus on physical-thermodynamic indicators for energy efficiency.

The term energy efficiency is often mixed with other terms, such as energy savings or energy productivity. While *energy savings* describes the composition of non-use of energy and energy efficiency (e.g. changing light bulbs and decreasing their duty cycle), *energy productivity* relates energy consumption to economic performance (e.g. turnover, value added). On the one hand, such ratios are certainly more meaningful than pure energy consumption figures. On the other hand, it is crucial to choose the right ratios so as not to create distorted results and enable targeted energy management [140].

[2] For example, shifting production into the nighttime might save energy costs, but increase personnel costs.

This shows that it is not trivial to accurately – and objectively – quantify energy efficiency. Issues arise when interpreting these indicators. For example, energy efficiency of a factory may decrease because of higher mechanization (and therefore higher energy use) rather than deteriorating technical efficiency. For economic indicators, changing energy or material prices over time might also lead to distorted interpretations [261]. The author of [211] provides an interesting discussion on the methodological issues in valuating energy efficiency many people are not aware of.

Nonetheless, what is important for our work is that one of the major ways[3] to increase energy efficiency is by decreasing the energy input E_{in}, see Equation (2.2) [186, p.15].

There are also other indicators that can be found in the literature. For example, the Specific Energy Consumption (SEC)

$$\text{SEC} = \frac{E_{\text{in}}}{\textit{Physical output}} \tag{2.3}$$

is an energy intensity measure and the inverse of the energy efficiency indicator. For unit manufacturing processes, the SEC is more favorable than energy efficiency as a Key Performance Indicator (KPI) [186] and is in this context often referred to as *Energy per Piece*[4] [139].

Combining energetic measures with physical units, like in Equation (2.2) and Equation (2.3), has the advantage that these figures can be objectively measured while also directly reflecting what consumers are actually requiring in terms of an end use service (e.g. Energy per Piece). This is why this kind of measure of energy efficiency is widely used in the industrial and commercial sector [186].

This has an interesting implication for simulation: When using simulation for evaluating energy efficiency in terms of these indicators, it is required that both energetic and physical (e.g. products) figures need to be considered, perhaps even be included in the dynamic simulation in order to dynamically align these figures and allow for more detailed evaluations. This motivates using a hybrid discrete/continuous simulation approach.

2.1.3 Improving Energy Efficiency in Industry

There are several studies in the literature which attest the industry a substantial energy saving potential, between 35% and 60%, depending on the sector [33]. The industry accounts for 30-40% of the overall primary energy consumption [267], and a large portion of this could potentially be saved by implementing effective energy efficiency measures [70].

Starting with a broad view on energy efficiency in industry, several fields of action have been identified in the literature, including [202, 261]

[3] The other possibility would of course be to maximize useful output by given energy input. However, we consider this out of scope for our work.

[4] This indicator is also of interest insofar as it can be interpreted as a kind of energy footprint of the product, similar to a CO_2 footprint. This has been elaborated further in [260].

- substituting renewable energy sources (wind, photovoltaics, hydropower, etc.),
- more efficient and low-emission power plants and energy conversion technology,
- more efficient industrial processes,
- energy recovery and energy reuse and
- improving planning and operation in production.

In the current work, we are concerned primarily with the last action, i.e. improving planning and operation of production facilities to save energy. This falls under the umbrella term energy Demand-Side Management (DSM), which includes all measures for energy management at the side of consumpation (as opposed to energy generation or distribution) [210, 250, 185].

When looking at where energy can be saved on the demand side, different levels of granularity are commonly distinguished [186, 261]:

- **Machine/Component Level:** This involves structural changes, such as using more efficient equipment and technology, as well as optimal choice of process parameters and improved process planning, such as avoiding idle times.
- **Production Line:** Across multiple production components, optimized production planning involves selecting the best possible process variants as well as scheduling products in an energy-optimal manner, e.g. avoiding peak loads or producing the most energy-intensive products at times when the most renewable energy is available.
- **Factory Level:** This includes construction and structural measures, like improving building insulation as well as operational aspects aside from the immediate production, like controlling auxiliary systems, Technical Building Services (TBS) (e.g. lighting) and Heating, Ventilation and Air Conditioning (HVAC).
- **Supply Chain:** Considering multiple factories, their interactions may be coordinated in a way that leads to an energetically optimal overall outcome on a global scale, resulting in a *industrial symbiosis* [87]. This might involve different factories within the supply chain as well as multiple production sites within a company. Transport logistics plays an important role, as does product life cycle analysis.

In the current work, we focus on the production line and factory level. At each one of these levels, energy is consumed and planning and operation decisions can be made that influence energy use [87]. It is difficult to quantify how much of the overall energy is attributed to which level, because it heavily depends on the industrial branch and the technologies involved. However, many high-technology companies have in common that a significant energy share – more than 50% in some cases – falls into areas outside of

the main production process, meaning production periphery such as TBS and HVAC. Traditionally, controlling the production periphery focused on availability and reliability – detailed coordination with the production is usually not considered [261]. Controllers typically regulate between fixed temperature thresholds instead of the control strategy being adaptive to the current production program. For example, it would be possible to adjust heat storages in anticipation of higher production utilization and a resulting increase in heat recovery [190]. This example underlines the importance of investigating the production facility as a whole for energy efficiency.

2.2 Production Planning and Control Systems

In general, Production Planning and Control (PPC) is concerned with the operative, temporal and quantitative planning, control and monitoring of all production processes that are necessary in the production of goods and commodities [302]. The goal of PPC in every company is to optimize the production system, by planning and managing the materials and capacities based on the customer needs. The temporal variations in demand, production and purchasing raw materials need to be coordinated in an optimal way through planning in order to make the production system work efficiently. Today, PPC forms the core area in operations management of every industrial production company [289].

Figure 2.2 shows the elements of PPC, summarized as a high-level process and inspired by the Aachen PPC model (see [248, 191] for more details) and other literature [158, 241, 111, 293, 323].

The overall PPC process consists of production planning and production control. While production planning generally involves planning resources, material and capacity as well as scheduling of future production jobs, production control is concerned with monitoring the current production execution. These tasks are usually complemented by Sales and Operations Planning (S&OP) and possibly further cross-sectional tasks, such as inventory management, order management, etc. [248].

The overall *planning* tasks take place on multiple levels with different time horizons[5]:

- **Long-term:** Starting from the top of Figure 2.2, S&OP involves production demand and sales planning based on customer orders, forecast projections and other market information – typically within the following year or longer. Long-term planning strategies are motivated by high pressures for high product availability and fast delivery in many supply chains against the background of increasingly volatile customer demand and pressure for cost and resource efficiency [293]. The result is a *demand plan* (*Dplan*) forecast determining which quantities of a given product should be available for delivery in which time period.

[5]The actual length of these time horizons strongly depends on the sector and types of products. For a typical job shop production, short-term planning covers around 1-2 weeks, while medium-term and long-term planning tasks often have time horizons of several months and up to one year, respectively [177].

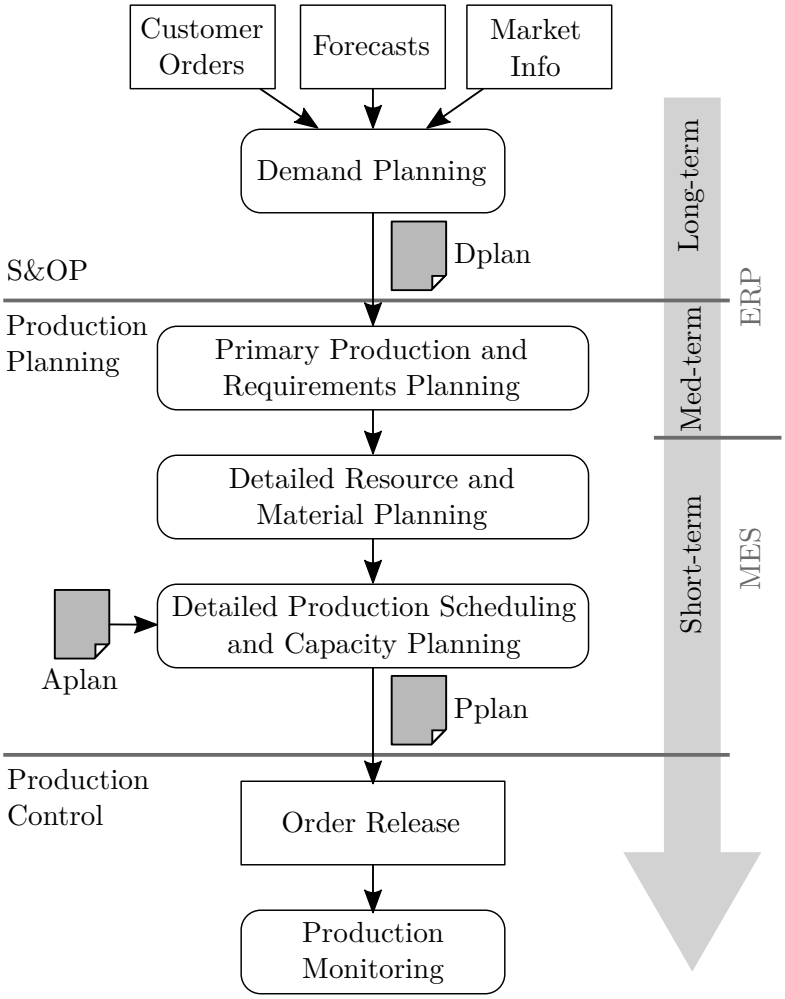


Figure 2.2: Process of PPC, complemented by S&OP. The figure also shows some of the relevant documents involved: demand plan (*Dplan*), production plan (*Pplan*) and work plan (*Aplan*).

- **Medium-term:** Based on the demand forecast, primary production and requirements planning tries to derive an aggregate resource and capacity plan on a medium-term horizon. The result is a rough production program under consideration of limited production capacity [248]. Depending on the complexity of the production process and the supply chain, primary production planning may be very simple or become more complex. In any case, rough process simulations are able to support medium-term planning, e.g. to identify capacity bottlenecks.

- **Short-term:** In the final stage, resources and material requirements are planned in detail on product level on a short-term horizon. The demand is calculated against warehouse stocks and decisions are made for either in-house production or external

procurement [248]. Detailed time scheduling and sequencing as well as allocation to machines and resources for the in-house production have to be planned based on a *work plan* (*Aplan*) that defines the process parameters (e.g. baking temperature and duration in an oven) and production procedure for each product type. This planning stage is also the most important one for energetic optimization, since scheduling (e.g. shifting start times into the night hours), sequencing (e.g. to exploit energetic synergies between jobs) and machine allocation (e.g. using an alternative machine that requires less energy) allows to exert the most influence on the energy demand during planning [261]. The result is a detailed *production plan* (*Pplan*) that specified which product is produced when on which machine(s).

After the planning is complete, the scheduled and allocated *jobs* can be released for production, at which point the *production control* takes over to monitor the production execution.

The described processes are usually supported by operative PPC software systems, which are an integral part of industrial Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES) systems [191]. These will be discussed in more detail in the next section.

2.3 Industrial Information Systems

The tasks involved in PPC are supported by computer-aided tools, which are part of the industrial information system within an automated production environment. The following provides a brief overview of these systems in order to be able to place the PPC systems in the proper context.

Traditionally, the system architecture of an industrial ICT system has been described as a pyramid structure [48], illustrated in Figure 2.3 and further formalized also in the IEC 62264 standard [144]. It is divided into five main functional levels with different IT components and communication technologies. Components communicate within their level (*horizontal*) as well as with neighboring levels (*vertical*).

Each level is responsible for its own tasks within an automated production. Depending on these tasks, specific technologies and communication standards have evolved that characterize the different levels.

The first level from below (*Field Level*) handles direct communication with the technical manufacturing process (which itself may be interpreted as being on Level 0) through sensors and actuators, which collect process data end execute commands for controlling energy and material flow [173]. It also contains the communication infrastructure on the field level traditionally consisting of diverse field bus protocols and industrial networks.

On the second level, i.e. the *Control Level*, the process data (e.g. temperature, switch positions) provided by the Field Level is used by local controllers to derive output actions to be sent back to actuators on the Field Level to control individual machines or facilities

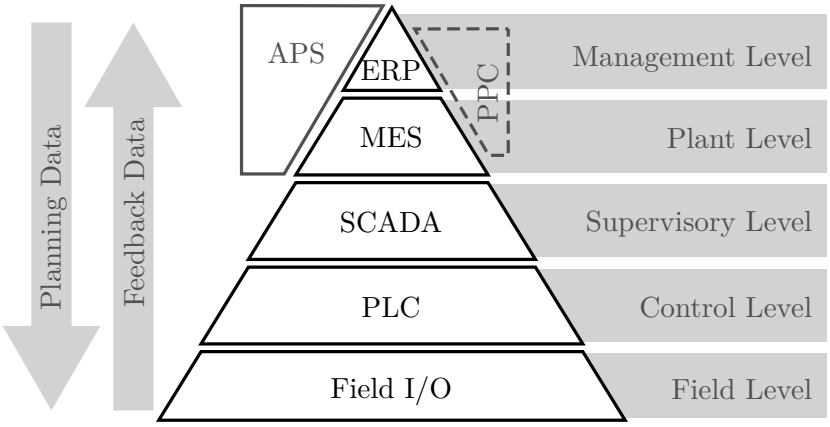


Figure 2.3: Typical automation pyramid of ICT systems in an industrial facility, including APS system and location of PPC tasks

according to specified set points and control targets. These local controllers are commonly implemented in Programmable Logic Controller (PLC) systems because of their flexibility, ease of use and robustness [297].

The main task of the *Supervisory Level* is to merge the resources from the levels from below via a suitable communication network with a host computer in order to obtain a holistic view of the production processes. A so-called Supervisory Control and Data Acquisition (SCADA) system are used as process control systems for monitoring and central control. SCADA systems collect and analyze real-time feedback data, such as status messages, but also energy consumption data from measuring devices, which they receive from the Control Level in order to prepare and display them to an operator and perhaps raise alarms if deviations occur. They also specify set points and targets for the Control Level by taking production instructions from the level above (i.e. the Plant Level) and deriving subtasks to be distributed to the controllers below.

On the *Plant Level*, all elements from the Supervisory Level are aggregated into a functional and organizational whole, called Manufacturing Execution System (MES) [316, 164]. It forms an interface between the technical control system on the lower levels and the business-oriented management level. The main tasks of the MES include material, operations, personnel and quality management and, in particular, detailed planing and control of the production. To achieve this, the primary planning data coming from the ERP system (see below) are broken down into individual production steps, and restrictions, such as machine and personnel capacities, are taken into account. The result of these planning activities is a detailed production plan, which is then transferred to the operative level below [164, 191]. From the Supervisory Level, (aggregated) feedback data, such as current order status, actual start and finishing times of jobs, machine data, and also aggregated energy consumption data, is received, which is collected and processed within the Production Data Acquisition (PDA) system. The PDA allows to identify deviations from the planned data and, if necessary, take appropriate control measures, as

well as to derive KPIs for the management level.

The top layer, i.e. the *Management Level*, is located on the global enterprise level and contains different computer applications, collectively called Enterprise Resource Planning (ERP) system. The ERP system collects in a central place all strategic and operational planning information as well as production feedback data from the lower layers relevant for enterprise management. The ERP level is responsible for all strategic tasks, such as order processing, finance and accounting or sales and distribution, as well as all long-term and medium-term production planning tasks as part of PPC, see also Figure 2.2.

The feedback data from the production (i.e. the technical process) moves through the levels from bottom to top and is successively aggregated – the planning information, on the other hand, goes from top to bottom and is successively translated into increasingly detailed control instructions. The pyramid shape comes from the number of components contained in each level. While the ERP system usually consists of a few centralized computer applications (processing large amounts of data), the bottom level may involve a large number of decentralized field components on the shop floor (each of them only processing a few data points).

While the traditional automation pyramid as a reference model emphasizes the strong hierarchical structure, the exact implementation of the levels may vary from company to company and often looks very different in other automation domains. Levels can sometimes be omitted or combined with other levels. In the literature, there are many different designs that differ in the number of levels and the exact names of the levels [192]. Some authors understand the Plant Level as being the combination of Supervisory Level and MES.

In recent years, modern industrial communication technologies in the context of Industry 4.0 [153, 38] and smart manufacturing [217, 179, 155, 75] have begun to increasingly flatten and soften the hierarchy of the automation pyramid by unifying the communication within the layers (horizontal integration) as well as across layers (vertical integration) [189]. Especially in the lower levels, the traditionally very heterogeneous infrastructures are increasingly being replaced by new technologies for industrial communication, e.g. Open Platform Communications (OPC) [147, 322] and more recently OPC Unified Architecture (OPC) and other Industrial Internet of Things (IIoT) technologies [321]. More information on this topic can be found in [150, 38, 217].

After presenting some of the background relevant for integrating simulation-based methods into industrial information systems, in the next chapter we discuss some related work, including hybrid co-simulation and other simulation-based methods.

CHAPTER 3

Related Work

In contrast to Chapter 2, where the basic terminology and relevant background have been presented, this chapter focuses on the state of the art on hybrid simulation and related fields. We discuss hybrid co-simulation in particular because of its stand as state of the art.

3.1 Literature Review on Simulation-based Optimization in Production

3.1.1 Optimization Methods

Since optimization in a complex solution space with simultaneous objectives is a difficult problem to solve, practical multi-objective optimization has spawned a wealth of approaches and solution methods [143, 90]. A general overview is given in [37, 308, 102], while [295] focuses on optimization in the food manufacturing industry. In general, combinatorial multi-objective solution methods can be categorized in [239]

- *exact methods* that guarantee finding an optimal solution, such as gradient methods [18], dynamic programming [23], or Mixed-Integer Programming (MIP) [218], and
- approximate/*heuristic methods*, including meta-heuristic, without guarantee that an optimal solution is found, such as greedy search [98], Simulated Annealing (SA) [290], Evolutionary Algorithms [255] or Variable Neighborhood Search (VNS) [127].

Usually, exact optimization is the method of choice if the optimization problem can be solved with effort that grows polynomially with the problem size. For NP-hard problems, however, where exact methods need exponential effort, even medium-sized

problem instances often become intractable using exact methods. In these cases, heuristic and meta-heuristic optimization methods can still provide a satisfactory solution. They often show good performance for many NP-complete problems and problems of practical relevance [239].

3.1.2 Meta-heuristics

While heuristic methods are problem-specific as they exploit properties of the problem and are typically derived from previous experiences with similar problems, meta-heuristics rely on a problem-independent high-level search strategy [317] to efficiently explore the solution space, often employing a combination of intensification and diversification mechanisms to escape from local optima and still perform a robust search [108]. As both are approximate approaches, they do not guarantee to find an optimal solution, however, they typically manage to find a feasible solution that is satisfactory for practical applications in a feasible time and with feasible computational effort. They are especially suited in cases when finding a global optimal solution is either impossible or impractical.

In recent years, research in the area of combinatorial optimization has shifted towards hybridization of meta-heuristics with other optimization techniques towards more problem-specific solutions [31]. These hybridizations include combinations of different meta-heuristics as well as combinations with exact algorithms (employed e.g. to solve specific sub-problems) and problem-specific heuristics. For example, a population-based Genetic Algorithm can serve as global search mechanism for exploration in combination with local search procedures during the intensification phase. In [310], a hybrid Evolutionary Algorithm is proposed in combination with Simulated Annealing for solving a multi-objective optimization. Similarly, [1] describes a hybridization with a multi-objective Tabu Search and a Genetic Algorithm.

In the context of industrial production planning, various heuristic and meta-heuristic solution methods have been proposed [156, 311, 2]. According to [295], meta-heuristic and customized multi-objective heuristic approaches are well-suited for applications in real-life industrial production planning problems (which typically are NP-hard), in contrast to exact approaches that require simplified models. Similarly, [311] compare different algorithms for near-optimal solutions after having encountered difficulties using an exact approach. Meta-heuristics allow to explore the search space more efficiently and effectively, especially if they are tailored to the individual problem [119]. Different meta-heuristic algorithms, such as Evolutionary Computation, Tabu Search, Particle Swarm Optimization (PSO) or Simulated Annealing (SA) have been successfully applied to various logistics optimization problems [253].

Besides *population-based* meta-heuristics, like Genetic Algorithms, Ant Colony Optimization [86] or Particle Swarm Optimization [80], which work with a population of candidate solutions to concurrently sample different regions of the solution space, *single-solution-based* methods, also called trajectory methods, iterate over a single solution are more exploitation-oriented and usually need fewer simulation evaluations, which improves over-

all computation time. Among these trajectory methods, Variable Neighborhood Search (VNS) algorithms have shown excellent capability for solving scheduling problems [238]. This is in accordance with other publications, e.g. [312, 3, 238], which have successfully applied VNS for job scheduling problems in the production domain. In [106], the authors compare different optimization methods for simulation-based optimization of production plans, in which VNS also leads to the best results.

3.1.3 Optimization of Energy Efficiency in Production

Numerous studies on the optimization of energy efficiency in production have been published in the literature. A comprehensive overview can be found in [235]. Some of the described methods focus on optimization without the use of simulation. The main strategies involve analyzing energy consumption in production and choosing energy-efficient process alternatives as well as addressing non-productive times including start-up and shutdown phases.

For example, the authors in [89] employ a graph-based approach in combination with Dijkstra and A* search algorithms for finding the most energy-efficient production state during unproductive times in production.

In [61], the authors investigate energy consumption reduction in production through effective control of machine start-up and shutdown schedules considering given productivity requirements and evaluating energy performance. They formulate a constrained optimization problem and discuss a greedy search algorithm for obtaining operation schedules.

Wang et al. [294] accomplish energy reduction in iron and steel batch production by using an integrated optimization model for optimal load scheduling and reduction of energy peaks. They formulate a non-linear optimization problem and introduce linearization techniques to derive a Mixed-Integer Linear Programming (MIP) model.

The authors of [84, 83] investigate the detailed energy consumption of production machines in different operating states in an effort to derive possible measures to reduce energy base load and optimize machine occupancy by addressing non-productive phases, such as standby, set-up and shutdown. Time and energy studies are carried out in different case studies to quantify the economical and environmental impact from a life cycle perspective.

Similar work has been conducted by Hacksteiner [122] and Dür et al. [169] at TU Wien. They aim to determine relevant energy efficiency and productivity Key Performance Indicators (KPIs) of machine tools based on measurement data for electricity and compressed air. The data are recorded via SCADA software and stored in a database. They divide the energy profile into base, process and peak load to determine energy efficiency and processing time.

All of these approaches have in common that they are based on measurement data and/or employ static modeling techniques for representing production systems. In the following, we will focus more on related works in the context of simulation.

3.1.4 Simulation-based Methods for Energy Efficiency in Production

An overview of approaches that employ simulation for improving energy efficiency in production systems is given in [261]. These methods agree that they see simulation as a promising tool that allows detailed modeling and analysis of complex systems [138] not only static, but dynamic over time, thus capturing the interactions between system components.

In the following, we will briefly discuss some of the more prominent related works that have been published in the literature. Many of them employ Discrete-Event (DE) simulation at its core, like the work of Thiede and Hermann [280, 139, 138, 282]. They extend the DE simulation with a simplified representation of the energy system and an evaluation module for optimization. The production machines are modeled with discrete operating states and associated power demands and time durations, which are stored in a production program that is executed and accumulates the energy input over time. The simulation model incorporates the production machines as well as Technical Building Services (TBS), and a separate PPC module handles production planning. The simulation is implemented in AnyLogic with additional Java code and evaluation and visualization in MS Excel. The TBS model is mostly based on algebraic equations for energy and flow balance. Initially, no Ordinary Differential Equations (ODEs) were used for continuous simulation of the TBS system. In later developments, the approach was extended to include a building model (in EnergyPlus) and a physical simulation of the machinery (in MATLAB/Simulink), as part of a multi-level co-simulation [247, 281].

A somewhat similar approach found in the literature is that of Junge [152]. His solution is also based on co-simulation and features simulation models for the material flow, energy flow as well as the building. The models are implemented in different simulation environments and coupled using a custom-built framework with a basic fixed-interval coupling strategy. The energetic simulation of the production facilities is based on interpolated measurement data, so no dynamic interactions can be taken into account. The production planning optimization is carried out via different heuristics taking into account energy efficiency. For example, energy-intensive products are scheduled at night, assuming lower temperatures during these hours. Critical parameters, such as lot size and threshold values, are determined using meta-heuristic parameter optimization. The method is applied on a case study of plastic injection moulding. A disadvantage of this co-simulation approach is its limited reusability. The co-simulation must be redesigned for new application cases and the model couplings have to be reimplemented. This requires considerable effort and knowledge.

We will go into more detail regarding co-simulation in Section 3.2, and in Section 3.3 we will discuss a more comprehensive co-simulation case study that has been developed by us in the course of a previous project.

Haag [120, 121] developed a concept to model the energy flows in all production facilities, including peripheral equipment. It uses a Discrete-Event Simulation (DES) and the modeling is based on machine states and power levels, which are stored in a database.

This applies to the main production processes as well as the peripheral equipment, which is divided into classes for modeling. The simulation is implemented in the software Plant Simulation, with parameters being read from MS Excel. The concept also includes a description of the planning optimization, which, however, was only carried out in a simplified case study. Potential extensions that are mentioned include modeling of the dynamic thermal behavior, the use of more sophisticated optimization algorithms and a more detailed modeling of the production processes and the periphery.

The approach described in [299, 300] is also based on DES and also uses state-based power levels for energy modeling. However, they extend this approach and define energy blocks for each machine type and process step, which are stored in databases and can be used for process modeling by combining them together to obtain the total energy demand. They call this the EnergyBlocks methodology. In addition to a more detailed modeling, this method has the main advantage of improved flexibility, since the energy blocks can be recombined and reused for new products. Auxiliary equipment, such as TBS, can also be included. However, this method requires a detailed preparation of energy profiles as energy blocks with corresponding plant states for different products. Furthermore, the energetic behavior can only be represented deterministically and dynamic interactions cannot be incorporated.

In [131], a multi-domain modeling approach for machine tools is described that aims at quantitative assessment of energy saving measures in metal-cutting production processes. It addresses electrical, mechanical as well as thermal aspects of machine tools, which are combined into a single dynamic simulation model. The implementation is based on object-oriented acausal modeling in MATLAB/Simscape [279] similar to Modelica [104]. A case study of a turning lathes serves as demonstration and for evaluation of the approach.

More strongly focused on optimization is the approach described in [228], which also uses DES for energy-oriented machine allocation planning. Various meta-heuristic optimization approaches are investigated and the Genetic Algorithm (GA) is identified as the most suitable one. The GA is extended by a hybridization with a local search procedure to accelerate the convergence. The method is applied to a case study in the field of textile production. The simulation is implemented in Plant Simulation with deterministic energy demands. The planning optimization focuses on a sequential planning of orders. As a simplification, only identical parallel plants are considered. A disadvantage of this approach is the long computation time of the GA optimization, which has been bound for reasons of practicability and at the expense of solution quality.

Römer and Strassburger [236, 234] investigate a hybrid simulation approach combining System Dynamics (SD), DES and Agent-Based Modeling (ABM) for energy efficiency analysis in production. They distinguish multiple modeling levels, ranging from individual processes via process chains up the the macro level computing the total energy consumption. The authors also argue that continuous modeling in combination with a discrete approach to map the material flows and logistic processes allows to show the complex interactions between material flow and energy usage in production closer to

reality. They apply the approach to a use case scenario in the mechanical processing of die-casting parts, where the simulation model is implemented in AnyLogic.

In [215], Peter and Wenzel also describe a co-simulation approach for hybrid simulation with bidirectional interaction between a DE material flow simulation and a Continuous-Time (CT) energy model using a communication middleware with TCP/IP interfaces. Plug-ins control data exchange, time synchronization as well as project management. They apply the approach on a case study from the automotive industry sector and demonstrate the interaction between production processes and energy flows through coupled simulation. They also discuss Key Performance Indicators (KPIs) for measuring the energy consumption for individual material flow items [301].

Schmidt and Pawletta [244, 212] also developed a hybrid simulation approach for describing resource consumption in industrial processes. Instead of using co-simulation, their approach is based on the Discrete-Event and Differential Equation System Specification (DEV&DESS) modeling formalism (see also Section 5.2) and the implementation is embedded into MATLAB/SimEvents, for which they develop a custom model library. Their components are divided into three main parts: the material flow aspect, the process physics model and a process control layer that maps the local process control operations of the component, modeled as state machines. As part of a case study, they describe a hybrid model of an industrial hardening furnace.

This simulation approach based on DEV&DESS allows a more integrated hybrid modeling on the component level compared to a co-simulation method and is similar to the method described in Chapter 5. However, their MATLAB/SimEvents implementation seems a bit cumbersome in some places and requires some workarounds due to the restrictions posed by the simulation environment. This in turn hinders scalability and general reusability of this solution.

3.1.5 Discourse

Some solutions combine simulation models with optimization techniques for systematically finding optimal scenario configurations. Due to the complexity of the system and the optimization problem, most of these techniques rely on heuristic or meta-heuristic methods.

The literature shows that most approaches employ DES with discrete state-based power levels that uses deterministic energy profiles and only allows very simplified consideration of dynamic energetic interdependencies. Similar approaches have also been published in [306, 242, 54, 54]. Most of them do not include production periphery or building facilities in the energy consumption.

Besides DES, co-simulation is also represented, which couples together more than one simulation model in different variants. Only few publications describe fully dynamic hybrid simulation with tight integration of material and energy flows that also captures their dynamic interactions [235]. However, most of these use simulation in a scenario-based manner without systematic meta-heuristic optimization.

3.2 Coupled Simulation

One intuitive approach to Modeling and Simulation of hybrid systems is to couple CT and DE simulators together and exchange data at runtime to form an overall hybrid simulation. While this approach has some obvious merits, it comes with significant challenges and drawbacks. To give an overview, this section presents the state of the art and some relevant background regarding coupled simulation. First, we discuss the concepts and terminology related to coupled modeling and simulation in general as well as co-simulation in particular. After that, we will discuss challenges of hybrid co-simulation as well as applications of co-simulation for energy simulation.

3.2.1 Methods for Simulation Coupling

Apart from *classical* modeling and simulation in a single Modeling and Simulation (M&S) environment, more and more approaches try to couple multiple equation solvers and/or multiple simulation methods or environments [50, 97, 125, 198]. In many cases, these approaches were born out of necessity to solve a particular simulation problem, for example coupling models of physical systems with controller algorithms for mechatronic systems [114, 198], or in Smart Grid simulation where power systems are coupled with communication network simulation [66]. In [130], a classification of coupling methods is given, presented in Table 3.1.

Table 3.1: Classification of methods for coupled simulation (adapted from [130])

	Monolithic simulation (single solution procedure)	**Distributed simulation** (multiple solution procedures)
Monolithic modeling (single modeling method)	I: Classical simulation	II: Model separation
Distributed Modeling (multiple modeling methods)	III: Model coupling	IV: Co-simulation

This classification can be applied to continuous systems[1] as well as hybrid discrete/continuous models, which is why it used general terms like *solution procedure* and *modeling method*. For a continuous model, a solution procedure would be a numerical ODE solver. For discrete-event models, the solution procedure involves the event scheduler as part of the simulation engine. Typical modeling methods are for example Ordinary Differential

[1] The authors of [291, 245] originally provided a similar classification for purely continuous models. The classification has been extended in [130] and is further generalized here.

Equation (ODE) modeling, System Dynamics (SD), Agent-Based Modeling (ABM), or DE.

Classical simulation (Quadrant I), i.e. modeling and simulation using a single simulation tool without coupling, is still the most common method. The simulation tool provides a modeling environment and a suitable model description language tailored to a particular domain. It can perhaps span multiple physical areas of engineering (e.g. Modelica) and often comes with libraries of pre-defined modeling elements that can be reused to build complex models in a time-efficient manner [291]. The overall model is then compiled to arrive at an executable simulation, during which a solution procedure (e.g. numerical algorithm) is used. This solution algorithm is typically provided by the simulation tool and tailored to work well in combination with the provided model description (e.g. DASSL in combination with Modelica).

When using a single modeling method, the overall model may still be divided up to be computed by multiple simulation algorithms. This is called *model separation* (Quadrant II). This can be employed for example to separate stiff equations for solving them with individual step sizes, so-called *multirate* methods [245, 270]. One notable example of a simulation tool for multirate simulation is MATLAB/Simscape, which allows to specify local solvers for isolated parts of the physical network [279, 131].

In contrast, when implementing individual model parts using different modeling tools, but a single solver, this method can be called *model coupling* (Quadrant III). The individual models are usually exported from the different tools to be imported and simulated in a common tool. The model export can be carried out by exporting equations or simulation code, either as symbolic equations, source code or compiled code [245]. In some cases, discretized equations are exported, effectively meaning that the numerical solver algorithm is also part of the exported model, which is why this case falls under co-simulation (Quadrant IV). One prominent example in this regard is the Functional Mock-up Interface (FMI) [30].

Co-simulation (Quadrant IV) uses multiple modeling methods and multiple solvers. In many cases, multi-domain or multi-disciplinary simulation models are divided into mono-disciplinary sub-models which are then implemented in specialized simulation environments [117]. One example of this approach is presented in Section 3.3 where a simulation model of a production facility is divided into a sub-model for the building, one for the energy infrastructure and one for the production machinery.

In [130], we provide a more precise definition of co-simulation:

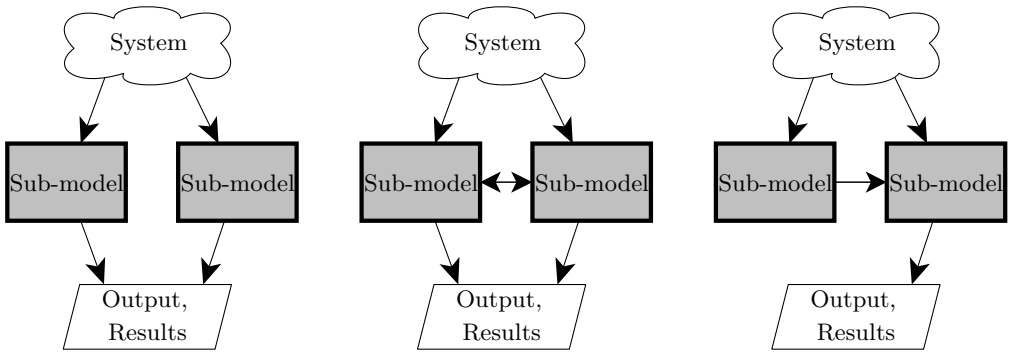
> *Co-simulation (cooperative simulation) is a method for simulating heterogeneous (continuous, discrete or hybrid) system models (typically instationary and time-dependent) by combining multiple sub-models and simulation algorithms (integrators, event schedulers, etc.) from different simulation environments,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

Combining multiple modeling methods and multiple simulation algorithms for co-simulation presents three immediate advantages compared to classic simulation [130, 291]:

- **Modeling advantage:** A coupled model may span multiple physical domains, and each of these domains can be modeled using languages, methods and tools especially suited for that particular domain. Domain experts are able to use tools they are already familiar with, which provide advanced user interfaces and modeling support, ultimately resulting in accelerated model development.
- **Simulation advantage:** Each domain sub-system may employ different computational algorithms (e.g. ODE solvers) tailored to the needs of the particular sub-model (e.g. step size, implicit methods for stiff systems), resulting in a more time-efficient co-simulation.
- **Engineering advantage:** Individual sub-models may be developed in parallel by different domain experts, thereby accelerating the model engineering process.

There are different ways to couple multiple sub-models to form a co-simulation. Figure 3.1 shows one possible classification of co-simulation couplings.

In the interfaced coupling, two (or more) sub-models may be run in parallel – and independently – with their outputs being combined (via an interface) afterwards. The integrated class incorporated continuous feedback and data exchange between the sub-models during runtime. In the sequential coupling, one of the sub-models has to be run first and its output is then fed to the next [271].



(a) Interfaced (b) Integrated (c) Sequential

Figure 3.1: Classification of co-simulation couplings, adapted from [271]

The bottom half of Table 3.1 is sometimes also referred to as *multi-method modeling* [110] or *multi-formalism modeling* [128]. This expresses that the sub-models use multiple different modeling methods or formalisms. It does not necessarily imply hybrid discrete/continuous modeling, since all of the employed methods may be of purely discrete (or continuous)

nature [159]. In practice, it is not always obvious which model falls into which category or which coupling is suited best for a given problem. The modeler has to decide what fits best and where the sub-model boundaries lie.

From a purely modeling point of view, the term *co-modeling* may also be appropriate to describe model coupling or co-simulation. However, this term is used more in the area of embedded systems [19], specifically hardware/software co-design, although it can sometimes also be found in connection with Cyber-Physical System (CPS) [42, 101].

3.2.2 Hybrid Co-Simulation

Hybrid co-simulation, as the name suggests, is a particular kind of co-simulation that combines CT and DE modeling methods in a non-trivial manner in order to model and simulate hybrid systems [5, 52]. The fundamental differences between these two paradigms lie at the heart of many of the challenges of hybrid co-simulation [113]. For example, while a CT state variable evolves continuously over time, meaning it is present at every time instant $t \in T$, a discrete-event signal may be absent at some time instants and assume multiple values at some other time instants (transiency) [67]. A DE simulation unit has to receive inputs and produce outputs often at the precise time some event occurs, while for CT co-simulation, a delayed response to the inputs is typical [113] (due to the orchestrator, e.g. using Jacobi-type coupling strategy, see also [130]).

A typical example of such a hybrid system is a thermostat regulating the temperature in a room [188], which is very similar to the controller used in our case study for an industrial oven, see Section 5.5. The continuous simulation unit represents a room (or a thermal mass in general) with temperature dynamics $T(t)$ including a source of heat P_h (e.g. a radiator):

$$\frac{dT}{dt} = \frac{1}{m \cdot c_p} \left(P_h \cdot k - UA \cdot T \right), \qquad T(0) = T_0, \tag{3.1}$$

where $m \cdot c_p$ is the thermal mass, UA the heat transfer coefficient and $k \in \{0, 1\}$ denotes the control input. The discrete simulation unit is a controller that controls (on/off) the heat source, depending on the current temperature sample T_i, which constitutes an input for this simulation unit. The thermostat is modeled as a state machine, shown in Figure 3.2.

Transitioning from one state to the other generates output events q with values 0 and 1, respectively. These values then have to be mapped to appropriate values of the input k for the continuous unit. The difference between q and k is that values for q only exist at times when the state of the thermostat changes, whereas k has to be able to be evaluated for any given time (i.e. continuous).

Obviously, these two simulation units cannot just be coupled together via output to input assignments. Any orchestrator for this co-simulation setup has to reconcile the differences between the continuous and discrete signals. This is what is referred to as semantic adaptation [81, 284].

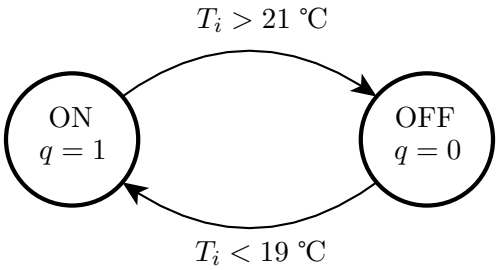


Figure 3.2: State machine model of a thermostat controller

The simplest (and most common) remedy is to use constant extrapolation, i.e. zero-order hold[2] function, to convert from discrete to continuous signals:

$$k(t) = \text{ZOH}(q). \tag{3.2}$$

On the other side, a sampling function is used to extract discrete events from the continuous function $T(t)$:

$$T_i = \text{sample}_i(T(t)). \tag{3.3}$$

Essentially, these two function create a wrapper around one of the simulation units to adapt signals to be compatible with the other unit. In general, two main approaches are distinguished in the literature for hybrid co-simulation orchestration, depending on where the wrapper is deployed [227, 113]:

- **Hybrid DE:** Every CT simulation unit is wrapped as a DE unit, for use with DE-based orchestration.
- **Hybrid CT:** Every DE simulation unit is wrapped to become a CT unit, for use with a CT-based orchestrator.

So, for the thermostat example, the hybrid DE approach would involve wrapping the continuous simulation unit as a DE unit, using Equation (3.2) and Equation (3.3), with a time advance that matches the size of the co-simulation step. As the hybrid DE approach is more common, plenty of other examples can be found in the literature [171, 204, 17, 201, 178, 99, 227, 313, 318].

The Quantized-State Systems (QSS) approach [166, 32], presented in Section 5.2.6, follows the same idea, but utilizes a different sampling than Equation (3.3), in particular state-based quantization instead of time-based sampling.

For hybrid CT co-simulation, both Equation (3.2) and Equation (3.3) would be used to wrap the thermostat controller into a CT simulation unit that takes as input the

[2] Zero-order hold is the term commonly used for digital-to-analog signal converters [148].

continuous temperature signal and outputs the continuous signal $k(t)$. Similar approaches can be found in [81, 96, 225]. The FMI standard for co-simulation [30] is one prominent example that employs the hybrid CT master algorithm for orchestration [276, 67].

Regardless of which approach is used, semantic adaptation between the CT and DE simulation units is a non-trivial task, especially if certain properties of the constituent systems have to be retained. Knowledge of the domain and the simulation units is paramount [113].

3.2.3 Challenges

As mentioned above, different issues arise when coupling multiple models, in particular (but not exclusively) in hybrid CT/DE couplings. The following gives a brief overview of some of the most significant challenges, in order to paint a picture of the current state of the art.

Semantic Adaptation: This issue was already discussed in Section 3.2.2. Semantic adaptations arise due to the fundamental need to integrate different Models of Computation (MoCs). While generic wrappers based on the underlying Models of Computation (MoCs) are possible [67], support for certain features (e.g. superdense time, rollback) depends on the capabilities of the simulation units involved. Whether a hybrid DE or hybrid CT approach for orchestration is more suited also depends on the current application and co-simulation setup. There is simply no best choice for all scenarios [113]. Even further, the manner in which events or signals are exchanged between units may need to be adapted at the technical level [284], for example changing how events are encoded. While wrapper-based adaptations of simulation units are perhaps the most intuitive solution, they introduce drawbacks of their own. Such a wrapper contains information that is encoded either directly in one of the sub-models (i.e. requiring semantic adaptations in the model for each co-simulation setup) or as part of the orchestration middleware (therby loosing flexibility) [113].

Adaptive Step Size: While an orchestrator with fixed communication interval [130] may be a simple solution, it is not the most efficient in terms of runtime. On the one hand, a fixed communication interval that is too small may lead to unnecessary overhead. On the other hand, events that occur between communication points experience a delayed propagation, thereby decreasing accuracy. More advanced communication strategies are needed in practice and several have been proposed in the literature [125, 36, 265, 99]. Still, accurately orchestrating discrete events and CT signals is still challenging, in particular in distributed co-simulation setups [16] and the fact that many advanced orchestration strategies require special features from the simulation units, e.g. rollback and resetting capabilities.

Determinism: Deterministic behavior should be maintained regardless of coupling. This either involves respecting the causality of events or ensuring that all possible

interleavings of executions always lead to the same result (i.e. confluence) [112].

Discontinuities: Considering for example a CT simulation unit that allows discontinuous state changes (e.g. Modelica [196]). If such a unit uses a DE wrapper, then the output signal of this wrapper, which is a series of time-stamped points, does not allow to discern a steep change of a continuous signal form a true discontinuity (in the CT model) [43, 182]. One possible remedy is to use extra information, e.g. an extra signal to mark the occurence and time of a discontinuity, as employed by the FMI [30].

Algebraic Loops and Illegitimacy: Resolving algebraic loops, i.e. non-causal dependencies between variables across simulation units, typically requires some fixpoint iteration technique, even in purely CT co-simulation. Not only does this add computational overhead, but the iteration may fail to converge. An orchestrator needs to be able to detect such a convergence failure as to not become stuck in an endless loop. An algebraic loop of a CT system is related to the illegitimacy property of DE systems [24]. An illegitimate model means that an infinite number of events occur within a finite time period [320, 166], thereby the simulation being unable to advance beyond this period. Illegitimate models hav to be recognized and appropriate measures have to be taken [113].

Stability: For hybrid or switched systems, it is possible that a particular sequence of events causes the system to become unstable, even if all the individual continuous modes of operation are stable [112]. Research is still required into the conditions under which a hybrid co-simulation system can become unstable, e.g. due to data quantization, change of orchestration or propagation delays.

Modular Composition: For large-scale systems, simulation units have to be composed modular and hierarchical [85]. Doing so involves multiple hierarchical orchestrators, which not only increases overhead but also brings the risk of loosing compositional properties like determinism or stability (see above). Constituent models should be able to be coupled to other models in different contexts in order to provide some kind of reusability. In co-simulation it is possible to get around the modularity aspect, but at a cost [113].

Standards for Hybrid Co-simulation: There is currently no well-established standard for hybrid co-simulation available. While there are the Functional Mock-up Interface (FMI) for CT co-simulation [30] and the High Level Architecture (HLA) standard for DE co-simulation [145, 71], both still have limitations for hybrid co-simulation. Extensions have been proposed in the literature, both for High Level Architecture (HLA) [16] as well as FMI [107, 276]. Broman et al. [43] proposed a set of test cases with their mathematically ideal unambiguous behavior, to test requirements for future hybrid co-simulation standards. Particular issues involve handling of simultaneous events, zero-width glitches and representation of time. They especially highlight the challenge involved in establishing a hybrid co-simulation standard: *"A standard that enables composition of simulation tools has two conflicting objectives. It needs to be sufficiently rigorous to define the*

meaning of a composition of components. And it needs to be flexible enough to embrace industry-standard and established simulators. The former demands a rigorous semantics, but the later creates pressure for less well-defined semantics." [43].

3.3 Co-Simulation for Energy Efficiency

There are multiple examples of using co-simulation for energy applications. Many of them revolve around research into smart grids [204, 187, 51, 85]. These applications typically combine power system simulation (continuous) with a model of the communication network (discrete), resulting in a hybrid model of a Cyber-Physical Energy System (CPS) [146].

The following presents a case study of a co-simulation for investigating energy efficiency in a production facility that we carried out previously as part of the research project INFO[3] [28, 137, 134, 169]. The case study involves co-simulating the dynamic inter-dependencies of production, energy supply network and building hull of a high-end metal-cutting production plant. The goal was to provide assessments of different energy saving measures and design variants during the planning process of the plant.

3.3.1 Reference Model

In order to coordinate systematic model development between the different sub-models of different engineering domains, in a first step a reference model was developed that formalizes and documents the overall system under consideration [183]. The reference model provides a generic description of a production plant with a focus on energy flows. It is intended to give the participating engineers an overview of the components the system is comprised of an, more importantly, their interfaces and interactions with other components, to guide concrete implementations of sub-models including corresponding communication interfaces and their subsequent coupling to a co-simulation [130, 169].

Figure 3.3 shows an overview of the developed reference model of a production facility as a network of 16 components with dynamic variable connections (black arrows) as well as static parameter dependencies (green arrows). The reference model includes general descriptions of components with their parameters and variables. Each component represents a distinct part of the overall system that can be found in most typical industrial production plants. The reference model includes planning components and parameter references that are not part of a dynamic simulation, but rather provide additional guidance for model parametrization based on planning data.

This generic reference model aims at providing a system overview and itself does not describe a concrete implementation of the internal model of individual components. Instead, it follows a black-box view of its components, thereby remaining independent of any modeling language or simulation tool. This allows for more flexibility for adapting

[3] http://projekt-info.org

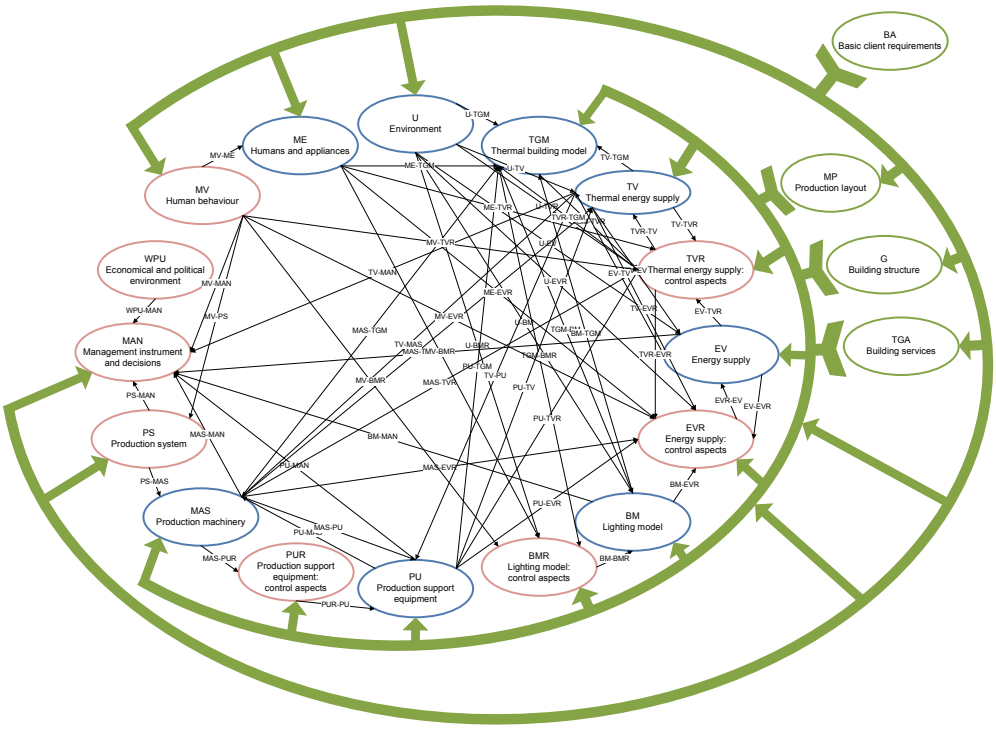


Figure 3.3: Reference model for developing simulations of production facilities. The model includes physical components (blue), information components (red) as well as planning components that provide static parameters (green). Black arrows show dynamic interactions between component variables. [130].

components to different specific implementations and requirements, in terms of model complexity, level of detail or data availability.

To derive a concrete model instantiation, individual components from the reference model may be pruned or condensed with other components to be encapsulated within a single simulation sub-model [183]. Internal variables have to be exposed according to the interfaces in the (condensed) reference model. The internal behavior can be implemented using different simulation environments (see below), including DE and CT dynamics, depending on which tool or modeling language is more suited for the sub-model at hand.

3.3.2 Co-simulation Implementation

After developing the reference model, the task was to apply it to a concrete case study of a production plant and implement an overall dynamic co-simulation. The overall co-simulation architecture is presented in Figure 3.4.

Three sub-models were developed in different simulation tools by different domain experts:

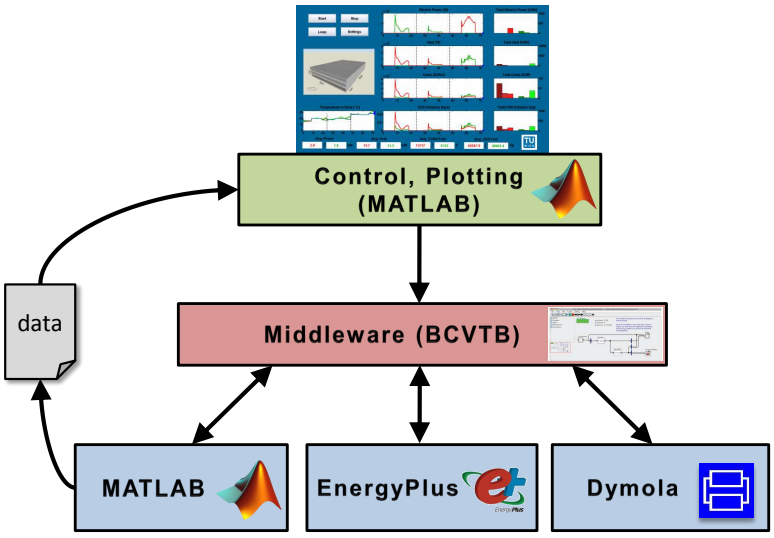


Figure 3.4: Framework for co-simulation between MATLAB, EnergyPlus and Dymola. The software middleware BCVTB handles synchronization and data exchange [130].

- Machines and production system (MATLAB/Excel): In order to be able to simulate larger time periods (e.g. one week, up to an entire year) as part of a co-simulation, it is necessary to employ a modeling approach for production machines that reduces model complexity by only considering the main energy flows relevant for the overall simulation. In this case, we opted for data-driven parametric models of discretized[4] load profiles with a significantly lower temporal resolution than physics-based high-fidelity models [131]. Detailed simulations together with measurement data can aid parameter calibration and validation [28] by providing high-resolution data, from which recurring operating states and corresponding energy levels can be identified for parameterizing the load profiles [134]. This approach was used to instantiate models for a pool of existing machine tools in the scope of the case study. Implementation was carried out in Excel[5] in combination with MATLAB [277]. The model computes the enery demand for the machines as well as waste heat, which serve as input for the energy system model and thermal building model, respectively.

- Energy system (Dymola): The energy system sub-model describes all Technical Building Servicess (TBSs), including equipment for supplying electric and thermal energy, like pumps, chillers and photovoltaic system. In order to compare different design variants [169], three different models of different energy systems have been implemented using Dymola [72] and the Modelica modeling language. To increase

[4]It is worth highlighting that this simplification results in a purely discrete time-driven model, as opposed to a continuous model.

[5]This was done due to practical advantages in the course of the project.

runtime efficiency, classical signal-flow-based modeling was employed instead of the acausal equation-oriented approach typically associated with Modelica [105], thereby leaving out unnecessary modeling details, reducing overhead and improving numerical stability [130].

- Thermal building model (EnergyPlus): The third sub-model depicts a thermal model of the building hull, including multiple thermal zones, that houses the production. The model incorporates weather conditions as well as time schedules for heating, cooling and lighting accounting for human occupation. The simulation receives heat gains from the production machines etc. and calculated heating and cooling demands for the energy system. The building sub-model was implemented as a Building Information Modeling (BIM) intance [88] and then exported for the simulation tool EnergyPlus [286].

These sub-models have to include additional interfaces necessary for co-simulation, which can be derived from the condensed reference model. At runtime, these sub-models interact by iteratively exchanging data via specialized software, so-called middleware, see Figure 3.4. For the case study, we used a prototypical open-source software framework, called Building Controls Virtual Test Bed (BCVTB) as middleware [304, 303]. The co-simulation follows a client-server architecture, where the BCVTB acts as the server for the simulator clients. The data exchange between client and server is carried out via a BSD socket interface for interprocess-communication using the TCP/IP protocol, which technically allows for the co-simulation to be run over a computer network [257]. The BCVTB employs a coupling strategy with fixed synchronization time step without iteration [304] – so-called Jacobi-type coupling [130].

Managing the co-simulation as well as gathering and processing the simulation results of the case study was also done in MATLAB, see Figure 3.4. MATLAB serves as a central point of contact for the user to execute the simulation as well as visualize the results in a graphical user interface [130].

Figure 3.6 presents the overall implementation in more detail, including the BCVTB graphical user interface. More details on this case study and its implementation can be found in [169, 130].

3.3.3 Results

The implemented co-simulation was used to compare different scenarios and parameter settings of the case study in order to investigate the impact of different design variants and energy saving measures on the overall energy efficiency [134]. A comparison between results of three different energy system variants is shown in Figure 3.5. It is intended to demonstrate the applicability of co-simulation for investigations into energy efficiency of production facilities.

The figure shows that Scenario 3 is the variant most suitable for the case study. In contrast, Scenario 1 is completely inept; a more detailed investigation uncovered that

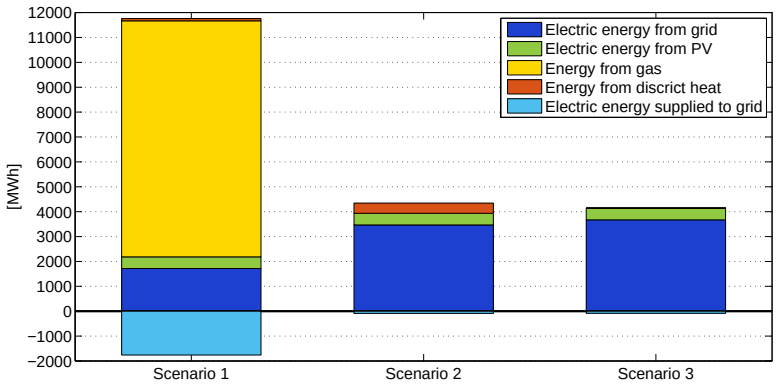


Figure 3.5: Comparison of annual final energy demand between three scenarios [130].

this was due to inefficient operation of the involved absorption chillers (that produce cooling energy for the building). For more details and further results, we refer to [169].

3.3.4 Discourse

Since the model is spanning several engineering domains (i.e. production machinery, TBS, building hull), a coupled simulation approach using different modeling tools and modeling formalisms seemed an intuitive choice. No single simulation tools was available that would have met all of the requirements and would have been able to fully model the complete system in the level of detail necessary to fully take advantage of the energetic interactions across the domains [130]. For the same reason, a mere interfaced or sequencel model coupling (see Figure 3.1) would not have been sufficient for this case study – only integrated coupling allows iterative data exchange at runtime.

As mentioned, the BCVTB software only allows a fixed synchronization time step without iteration, i.e. Jacobi-type coupling. Although this is the easiest coupling strategy to implement[6], it comes with significant restrictions [130]:

- Input variables have to be extrapolated between macro-steps, which increases numerical errors and may even cause stability problems.
- When using a DE simulator client, events are not propagated immediately, but only at the next synchronization point. This again increases errors and may lead to unintended behavior. This makes BCVTB not well-suited for event-driven sub-models.

Another drawback of the BCVTB software is its low-level data handling. Variables have to be manually prepared and combined into a single data vector before being transmitted.

[6] Perhaps the biggest advantage of this simple coupling is that, since there is no repetition of macro-steps, no external resetting of simulator clients is necessary, which not all simulation tools allow to do.

Not only is this cumbersome and error-prone, but also the semantics of data variables gets lost. The data semantics is not fully specified in the reference model – instead, the domain experts as well es the co-simulation engineer have to agree on common semantics during implementation.

The reason this case study is shown here to this extent is because it later inspired the work presented in the following chapters. The reference model in Figure 3.3 is similar to the metamodel developed in the following (see Chapter 7), although less formalized. They both have in common that their intention is to guide model development across different domains by unifying component and interface descriptions. They ultimately allow for better and more seamless integration between components of different domains and modeling formalisms [130].

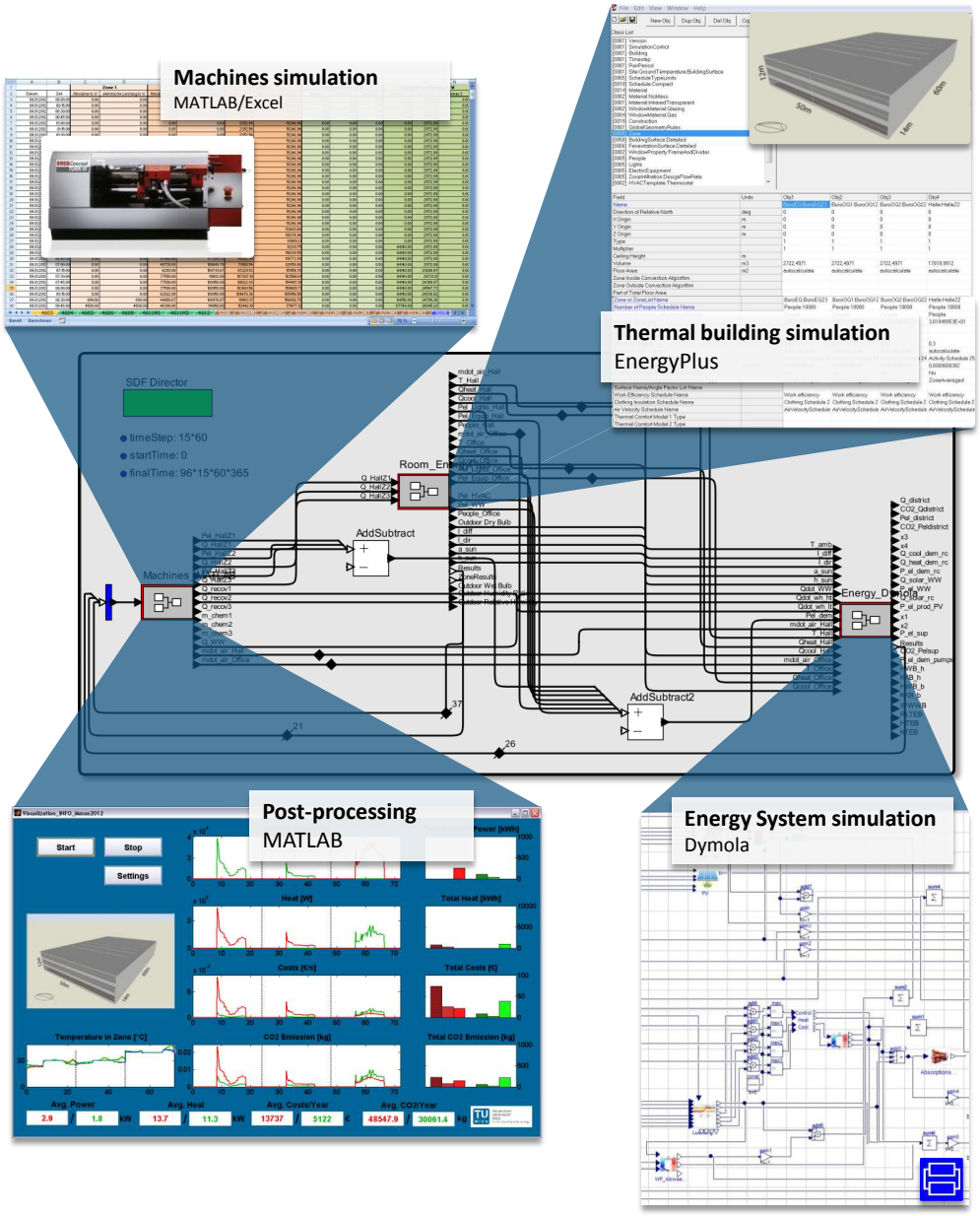


Figure 3.6: Overall co-simulation implementation for the case study, including MATLAB, Dymola and EnergyPlus simulations, coupled via BCVTB middleware [130].

CHAPTER 4

Concept for Simulation-Based Energy-Aware PPC

In this chapter, we want to discuss how energy aspects can be integrated into modern Production Planning and Control (PPC) and Advanced Planning and Scheduling (APS) systems by means of model and simulation-based methods. We discuss the general architecture within the automation system and how the associated data are handled. We then present an approach for conceptual component-based modeling of production systems that builds the foundation for the hybrid simulation. The chapter concludes with an overview of the general engineering workflow for developing component-based simulation models for practical applications.

This also outlines the main contributions of this work presented in the subsequent chapters: A method for modular hybrid modeling of production systems, a meta-heuristic optimization procedure for energy-aware production scheduling, and a formalized workflow for engineering application models.

4.1 Integrating Energy into PPC

The idea is to deploy a computer-aided planning module that is part of a PPC or Advanced Planning and Scheduling (APS) system and provides a simulation component for energetic assessment of different planning scenarios, see Figure 4.1. An optimization component is able to use this energetic assessment and evaluate it as part of a multi-objective optimization problem together with other production targets. However, the goal is not to replace an entire industry-grade PPC system with all its data infrastructure, but rather to exploit this infrastructure and extend it by an energy-aware planning tool. This way, the planning module can be easily connected to the automation system in order to obtain real-time data for parametrization.

To provide simulation-based functionality, it is necessary to engineer an application model of the production system in a suitable development environment. This can be a programming environment, a modeling tool or a model editor that is tailored to the task of modeling production systems. A more detailed discussion on this is provided in Section 4.4.

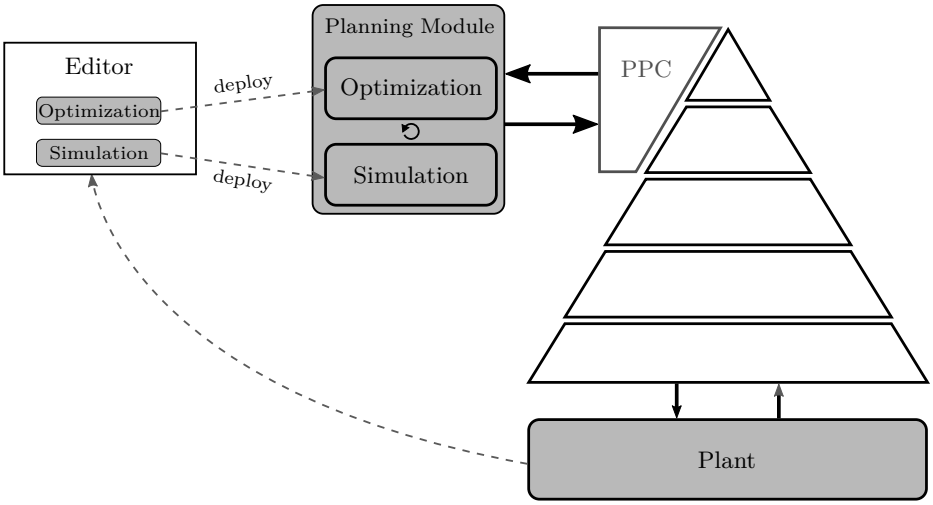


Figure 4.1: Simulation-based planning module as part of a PPC system in the automation pyramid architecture

During operation, a planning task is triggered periodically, e.g. once a week, once a day, or more often, depending on the planning horizon. The PPC system holds the current production program that is to be optimized. It transfers it to the planning module and triggers the optimization task, which iteratively finds an optimal value. Thereby, the simulation is used to evaluate different scenarios and planning variants based on a given target function. Additional data from the PPC may also be included in the target function, e.g. data on the current energy price, or shift schedules of the working personnel. The final result is then returned to the PPC, which displays it to the human operator as decision support and planning proposal.

The human operator is also required to provide the optimization target, i.e. which additional part goals are to be included and how they are weighed against each other.

In order to provide real-life tangible, the simulation has to be parametrized based on real data. This applies especially to the initial values, which, in contrast to static parameter values, may be different for each planning run and ideally reflect the current state of the real production system. The data for the initial values can ideally be taken directly from the automation system, allowing them to always be up to date. However, this requires adequate automation infrastructure and custom engineering. Providing such data connections is not within the scope of this work and is therefore excluded in the following.

As mentioned, the planning module itself contains an optimization and a simulation component, the exact interaction of which is explained in more detail in Chapter 6. However, it is worth noting that it may make sense for certain applications to use the simulation alone, without the optimization component, in order to manually evaluate individual production scenarios and obtain predictions about future energy consumption.

4.2 Conceptual Modeling for Simulation

Conceptual modeling in general is a process that elicits the general knowledge about a problem domain and comprises it into a Conceptual Model (CM) in order to develop a solution for a given problem [205]. It involves the abstraction of a model from a real or proposed system [232]. This can of course be applied to simulation studies as well, where conceptual modeling shows the general knowledge about what is going to be developed in a simulation model [233, 55]. A simulation CM is an abstract representation of a system describing its elements, relationships, boundaries and assumptions without reference to specific implementation details [100, 55]. A CM provides a means of communication between the stakeholders in a project and avoids ambiguities. They can be specified in a variety of communicative forms, such as diagrams, drawings, graphs, equations, images or text [20].

Usually, a CM starts off as an informal (mental) model, which is gradually refined and formalized to arrive at a description that can be implemented into a simulation model. Conceptual modeling thereby marks the transition from analyzing the problem space to system design in the solution space [40]. A CM in itself cannot be executed directly [55], since it typically does not define execution semantics.

Transformation from the CM to an executable implementation is either done manually by Software Engineering experts, or, depending on the level of model formalization, may be automated by specifying transformation rules, techniques or patterns on the model. This, however, requires a higher degree of formalization in the CM, thus shifting some of the workload from implementation to modeling.

4.2.1 Frameworks, Tools and Languages for Conceptual Modeling

Research on simulation conceptual modeling has increased in recent years since conceptual modeling traditionally being seen as a process that is almost completely performed casually [55]. Different frameworks for conceptual modeling have been proposed in the literature, some more formal than others. Robinson [230, 231] describes a conceptual modeling framework consisting of five iterative activities: understanding the problem situation, determining the modeling and general project objectives, identifying the model outputs, identifying the model inputs, and determining the model content (scope and level of detail). In [170], the authors recommend the use of soft systems methodology [60] in undertaking knowledge acquisition and model abstraction and they provide examples

in discrete-event simulation [55]. Different conceptual modeling frameworks for simulation are presented in [274, 27].

Most of these frameworks include two sub-stages, which may be executed in parallel: system structure definition and abstract behavior definition. This is commonly achieved in conceptual modeling by combining different diagramming techniques [162].

Especially with regard to software engineering, the diagrams of UML [207, 118] are very well suited for representing system structure and behavior, but also SysML is becoming increasingly popular [141]. Besides such general-purpose modeling tools, various domain-specific conceptual modeling languages have been defined for different simulation domains, such as the Simulation Modeling Language (SimML) [6], Simulation Model Portability Standard 2 (SMP2) [314] or BPMN [92], among others. In addition, ontologies also provide a suitable means for simulation conceptual modeling, as suggested in [193, 254].

Although visual diagrams of some formalisms, such as Petri Nets [216] or DEVS [320], are also used for conceptual modeling, despite the fact that these formalisms are not designed as conceptual modeling languages, it requires that all of the stakeholders involved are familiar with the formalism. The potential advantage of this approach is that the CMs can be refined incrementally until the final full model basically becomes the simulation model. However, in most cases, problem owners are not familiar with such a specialized language and the CM is therefore difficult to communicate during the conceptual modeling stage [55]. For practical applications, it is critical that the M&S expert and the software engineer as well as the problem owner (e.g. customer) can understand the conceptual model so that they can agree on it.

4.2.2 The Model Continuity Problem

As mentioned, CMs are not intended to be executed and thus provide no formal execution semantics, in contrast to simulation modeling languages that do need to be executed. A model continuity problem arises when there is no formal alignment between the models when different modeling languages are used at different stages [55]. This puts the responsibility in the hands of the simulation model programmer to ensure correctness and quality of the model implementation. Although there are many languages and tools for simulation conceptual modeling, special attention has to be put on semantic model alignment to mitigate the problem of model continuity. In the next section, we will present a concept for domain-specific model alignment in component-based system modeling. Some approaches that take the idea of model continuity further by providing more formal and precise semantics that enable the explicit use of the conceptual models using formal transformations into an executable simulation [209].

4.2.3 Component-based Modeling

For developing and implementing simulation models of dynamic systems, many software tools follow a component-based paradigm [68]. In this bottom-up approach, well-defined model components encapsulate certain internal dynamics, which, when being composed

with other components into larger models, together describe the overall model behavior. The components interact with one another only via specified interfaces – they usually cannot influence each other's internal states directly. More precisely, in [44], the authors define a component (in a rather broad, general and software-oriented notion) as

> *an independently deliverable piece of functionality providing access to its services through interfaces.*

Component-based modeling facilitates *modularity* and *separation of concerns* for managing the complexity of large-scale models [133]. From a workflow perspective, it also allows to distribute model development among different development experts. Such a distribution may take place along the domain boundaries (e.g. production, building, energy system), as well as along the level of application, from application engineers, who develop specific applications, to software engineers that provide the necessary computer-aided engineering tools. Modeling and Simulation (M&S) experts can create libraries of model components that are validated, trustworthy and well-documented. These components can be instantiated for different applications and in different contexts, thereby facilitating model *reuse* (in particular *black-box* reuse). Reusing model components is crucial in an attempt to reduce the effort and costs necessary for developing new application models [58, 68]. The importance of component-based development lies in its efficiency [79].

However, in order to retain modularity and composability, it is necessary to encapsulate all aspects of a particular component within uniform component boundaries and adhere to specified interface semantics. This can present a challenge in the context of hybrid simulation, where discrete and continuous model aspects have to be combined in a modular and runtime-efficient manner [133].

When implementing a component-based modeling paradigm in software, it is intuitive to employ traditional Object-Oriented Programming (OOP) [46, 298]. This way, components are implemented as classes which can be organized in a library of model components to be instantiated and configured for different situations [183]. While both, component-based modeling and Object-Oriented Programming (OOP) seem quite similar, they do have certain differences [44, 269, 223]. Nevertheless, OOP has proven to have significant advantages in software engineering and it is common enough that most software engineers nowadays are familiar with it.

4.3 The Cube Concept

In an effort to manage the complexity during the analysis of real-world interdisciplinary production systems, an approach for conceptual modeling is developed that facilitates abstraction as well as communication between stakeholders. It encompasses the view of different engineering domains (production, energy system, building, etc.) in a unified concept.

This approach divides the overall system from an energetic point of view into well-defined manageable modules, called Cubes, which then allow a focused system analysis and modeling independent from the surrounding environment [130]. A Cube may for example be a baking oven, a machine, an electric chiller or a room within a building. Figure 4.2 depicts an example configuration of Cubes dividing a production facility in a hierarchical manner.

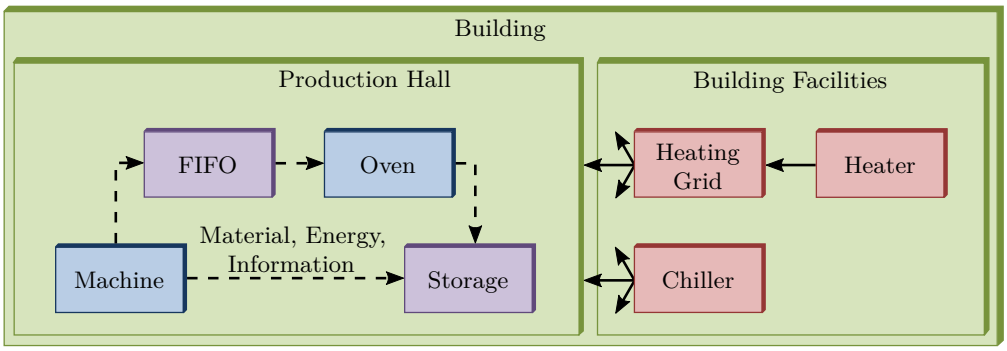


Figure 4.2: Example of a production facility consisting of different Cubes

A Cube represents a real-world object that comprises a well-defined internal behavior and interacts with its surroundings by exchanging energy, material and information flows. The interfaces are defined on an abstract level in order to ensure interoperability and applicability of the Cube concept on a variety of application cases as well as engineering domains. In the current case, four different domains are distinguished: machines and production processes, logistics, technical building services (i.e. energy system), and the building. More details are given in [168, 130].

The Cube concept serves three main purposes: It allows to manage the complexity of large-scale applications by divide and conquer, it facilitates communication between stakeholders of different domains as well as the problem owner, and it defines a domain-specific specialization of the component concept. Cube align intuitively with component-based conceptual modeling, where models are built from well-defined components. In the same way, Cubes can serve as building blocks in a simulation model [130].

From a modeling perspective, Cubes can incorporate discrete as well as continuous behavior, in the form of material and energy flow. The energetic behavior can be modeled by drawing energy balances around the borders and derive corresponding dynamic balance equations. This way, transient dynamic behavior can be incorporated in order to analyze time-dependent energy exchange between Cubes. For the discrete dynamics, entity-based structure in combination with discrete events allow to simulate persistent and traceable products (e.g. work pieces) and can intuitively be described using state diagrams. These discrete and continuous aspects are often tightly intertwined and interfere with each other. For example, the internal temperature of a baking oven needs to reach a certain point before entities can be processed [130].

In order to be able to build hybrid simulation models from Cubes, it is necessary to implement them in a modular manner and provide them as part of high-level conceptual modeling libraries [58]. For this reason, we follow a modular hybrid simulation approach based on the Discrete-Event System Specification (DEVS), which is described in more detail in Chapter 5.

The Cube concept was developed in the course of the research project Balanced Manufacturing (BaMa) and is also described in related publications, e.g. [183, 256, 219, 261].

4.4 Engineering Workflow

Engineering planning modules for new applications, and in particular developing the required simulation models based on Cubes follows the general workflow depicted in Figure 4.3. The overall development process can be divided into different roles. As part of a domain engineering process, a model engineer (who is typically a M&S expert) develops conceptual models of Cube components that occur in the considered domain and records them in the form of a Cube specification. This Cube specification can be kept semi-formal and serves as the basis for communication with the software engineer. It can employ different description tools, such as diagrams, formulas or even textual descriptions. Based on this specification, a software engineer can implement the Cube as a model and make it available in a repository.

During the application engineering process, an system engineer analyzes the production system at hand and models it by taking Cube models from the repository and instantiating, interconnecting and parameterizing them. In case Cubes are missing, they have to be created in consultation with the model and software engineers in order to extend the Cube library. The result of the application modeling task is a coupling specification, which can be formalized in varying degrees, depending on the subsequent task where a software engineer completes the implementation. More details are given below. In addition to specifying the simulation model, the optimization component can also be instantiated and calibrated to obtain a fully implemented planning module.

Depending on the scale of the project, some of the developer roles may of course coincide, to the point where all roles are fulfilled by a single person. However, for large projects, it usually makes sense to split up the development process.

For the concrete model implementation – be it application model or Cube specification – different approaches can be chosen, which are compared in Figure 4.4. In the traditional specification-driven paradigm, the model is recorded in a suitable manner (e.g. using SysML diagrams), to be implemented manually by a software engineer. For Cube-based application models, it is intuitive to implement them using an Object-Oriented Programming (OOP) language [183]. However, this approach has the disadvantage that the conceptual gap between problem specification and implementation is rather large, which can quickly lead to misunderstandings and errors in the model. This can be

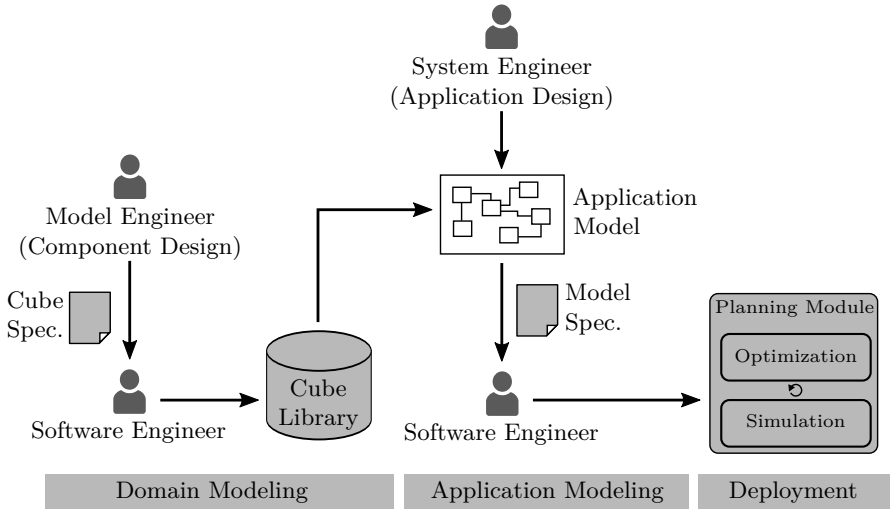


Figure 4.3: Typical engineering workflow with domain engineering done by model engineers and application engineering done by system engineers, both of which are supported in the implementation by software engineers.

remedied by using a formalism-driven approach, for example based on DEVS, which provides a formally sound infrastructure for implementing models [219]. This ensures that the implementation stays consistent and that the behavior is always transparent. On top of that, an additional layer for model-driven development may be introduced, in order to specify models on a higher level and afterwards transform them into low-level implementations.

In Chapter 5, we follow the formalism-driven approach for implementing Cube models, whereas in Chapter 7, we investigate the mode-driven engineering paradigm.

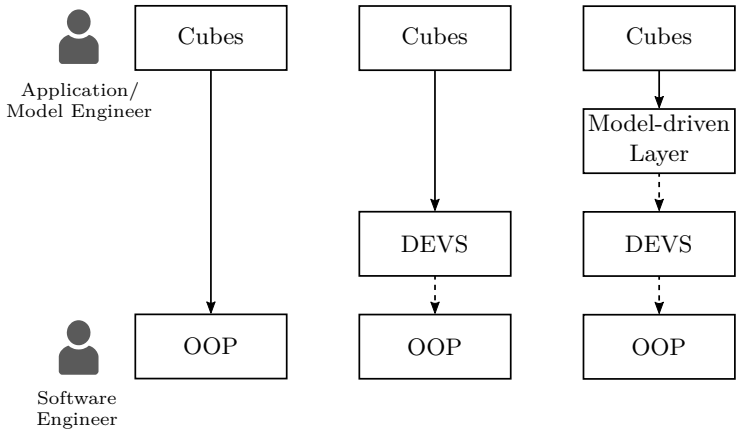


Figure 4.4: Different model engineering paradigms: specification-driven (left), formalism-driven (middle) and model-driven (right).

In summary, the overall concept can be divided into three main modules, depicted in Figure 4.5. These modules build on one another with the latter modules being optional in its implementation. For example, simulation may be used independently from the optimization to predict energy consumption in certain scenarios. The results can then be fed into an existing PPC/APS system, which takes over the systematic optimization. Similarly, it is also possible to implement simulation models with or without the Model Engineering layer, as already illustrated in Figure 4.4.

This optional character is also the reason why, in the following chapters where we describe the modules individually, we start at the "bottom" with the simulation being presented in Chapter 5, then build the simulation-based optimization method on top in Chapter 6 and finish off with investigating the engineering aspects of both parts in Chapter 7.

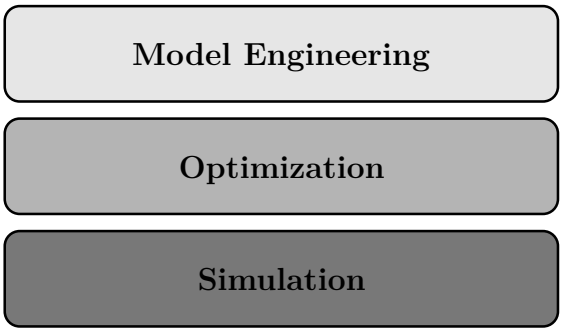


Figure 4.5: The three main modules of the overall concept: Simulation, Optimization and Model Engineering. The parts build on one another where the latter modules might be optional.

CHAPTER 5

Hybrid Simulation of Production Systems

In this chapter, we dive into the method for hybrid simulation of production systems that has been developed in the course of the research projects. We first focus on the formalism-driven model development paradigm (cf. Figure 4.4) using semi-formal model descriptions. We investigate the model-driven development paradigm in Chapter 7. The described method has been applied in five case studies altogether, some of which are being presented in the second part of this chapter.

5.1 Introduction

We have already elaborated on why employing hybrid discrete/continuous simulation is important in the context of interdisciplinary energy assessment in industrial production. It allows both the material flow to be modeled as Discrete-Event Simulation (DES) and the energy flow by means of differential equations, while also taking into account dynamic interactions between these domains. Continuous representation of energy flow, as opposed to discrete energy profiles, enables to accurately incorporate transient dynamics, for example the heat-up process of an oven or the thermal heat capacity of the building.

The most common approach for hybrid simulation, not only in an industrial context, is co-simulation, in which multiple existing simulation tools (typically one discrete and one continuous) are coupled together to exchange data at runtime. While this approach is the fastest one to implement and is able to make use of established simulators, maintaining and reusing such models is usually very difficult and often impossible because of the low-level model coupling. Also, event-accurate synchronization and efficient communication is not trivial for hybrid co-simulation [266] and requires sophisticated coupling mechanisms that have to be supported by the different simulation tools involved.

We pursue a different approach for hybrid simulation that is based on a system-theoretic description, called Hybrid PDEVS (PDEVS). Compared to co-simulation, this approach promises several interesting advantages, such as tighter integration of hybrid aspects and better performance, but most importantly it offers improved reusability of model components, which is crucial when developing complex large-scale simulation models. This has also been investigated in [130].

The remainder of this chapter is structured as follows: In the next section, we want to give an introduction into the hyPDEVS specification for dynamic system simulation and discuss some of its properties as well as simulation execution and approaches for handling the ODEs. After that, Section 5.3 elaborates on the Cube concept in the context of simulation. We present the developed Cube model library and discuss in Section 5.5 an example in detail. Section 5.6 discusses the simulator implementations, while Section 5.7 and Section 5.8 present case studies where the developed Cube models have been applied. We then conclude with a short discourse.

5.2 The hyPDEVS Formalism

As already mentioned, we employ a formal model description, called hyPDEVS, for hybrid simulation. The following provides a brief introduction into the formalism in order to give the necessary background.

The classic Discrete-Event System Specification (DEVS) is a formal model description language for Modeling and Simulation (M&S) of Discrete-Event (DE) systems. DEVS is based on systems theory and was first introduced by Zeigler [320]. It provides a formal syntax accompanied by an abstract simulator algorithm to specify operational semantics on how to execute these models. Based on DEVS, a family of extensions has been proposed, including Parallel DEVS (PDEVS) with improvements for handling concurrent events [64], Stochastic DEVS (STDEVS) introducing stochastic features, and Discrete-Event and Differential Equation System Specification (DEV&DESS) combining the description of DE and Continuous-Time (CT) systems [220, 129].

Also aiming at hybrid discrete/continuous systems, another extension, which we will denote with hyPDEVS in the following, was first introduced by Deatcu [77] and is similar to the DEV&DESS formalism by Prähofer [221, 220], with the difference being that it allows improved handling of parallel and concurrent events (since it is based on the PDEVS extension [64]). After evaluating both methods with regard to their capabilities to model hybrid systems in an industrial context [222], it turned out that DEV&DESS was unfit for our applications, due to its shortcomings related to Classic DEVS, and we instead had to opt for a solution based on Parallel DEVS (PDEVS), i.e. hyPDEVS. The drawback, however, is that hyPDEVS is less known in academia and is lacking off-the-shelf tool support [229].

5.2.1 hyPDEVS Atomic

All these DEVS-based formalisms allow to build models from components in a hierarchical manner by distinguishing between *atomic* and *coupled* components. More formally, a hyPDEVS *atomic* is specified by the tuple [77, 130]

$$A_{\text{hyPDEVS}} = \left\langle X^d, X^c, Y^d, Y^c, S, f, c_{se}, \lambda^c, \delta_{\text{state}}, \delta_{\text{ext}}, \delta_{\text{int}}, \delta_{\text{conf}}, \lambda^d, ta \right\rangle, \tag{5.1}$$

where

X^d is the set of discrete event input values,

Y^d is the set of discrete event output values,

$X^c = \{(x_1^c, x_2^c, \ldots, x_k^c) | x_1^c \in X_1^c, x_2^c \in X_2^c, \ldots, x_k^c \in X_k^c\}$ is the structured set of continuous value inputs with input variables x_i^c,

$Y^c = \{(y_1^c, y_2^c, \ldots, y_l^c) | y_1^c \in Y_1^c, y_2^c \in Y_2^c, \ldots, y_l^c \in Y_l^c\}$ is the structured set of continuous value outputs with output variables y_i^c,

$S = S^d \times S^c$ is the set of discrete and continuous states,

$f : Q \times X^c \to S^c$ is the rate of change function,

$\lambda^c : Q \times X^c \to Y^c$ is the continuous output function,

$c_{se} : S^c \to S^c$ is the state event condition function,

$\delta_{\text{state}} : Q \times X^c \to S$ is the state event transition function,

$\delta_{\text{ext}} : Q \times X^d \to S$ is the external state transition function,

$\delta_{\text{int}} : S \to S$ is the internal state transition function,

$\delta_{\text{conf}} : Q \times X^d \to S$ is the confluent transition function,

$\lambda^d : S \to Y^d$ is the discrete output function,

$ta : S \to \mathbb{R}_0^+ \cup \{\infty\}$ is the time advance function,

$Q = \{(s, e) | s \in S, e \in [0, ta(s)]\}$ is the set of total states.

The variable e constitutes the elapsed time since entering the state s. The specification differentiates between discrete (X^d, Y^d) and continuous (X^c, Y^c) input/output values (X^c and Y^c have to be real vector spaces) as well as states (S^c, S^d). These sets define all the possible values the respective variables can adopt. While discrete states in S^d describe dynamics that only changes value at discrete points in time (called events), S^c may change continuously over time.

Figure 5.1 illustrates this modeling concept of combining discrete and continuous inputs, outputs and states. Input ports of X^d accept events and input ports of X^c accept piecewise continuous segments. The discrete and continuous parts can influence each other's states, and each part produces corresponding outputs Y^d and Y^d, respectively [319]. While the event input only influences discrete states in S^d, the input X^c can influence both model parts by means of state events. The continuous part can cause events to occur whenever

a condition on the continuous elements becomes true, specified by means of the function c_{se}. Such a condition can typically be viewed as a continuous variable crossing a certain threshold or two continuous variables meeting (in which case their difference crosses zero). In such a situation, an event is triggered and the state is changed discontinuously, which is carried out by computing the state event transition function δ_{state} [130]. These events are called state events, in distinction to internal events (i.e. time-driven) and external events (from inputs X^d).

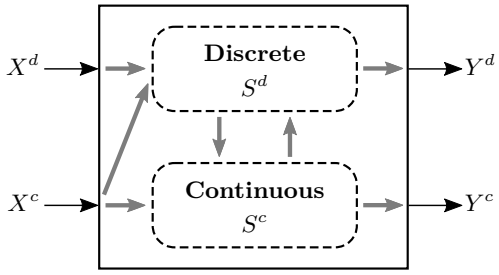


Figure 5.1: hyPDEVS combined model with discrete and continuous inputs, outputs and states

For the continuous dynamics, the rate of change function f in Equation (5.1) defines the model equations in terms of ODEs, and λ^c computes the continuous outputs. One notable characteristic of λ^c is that it may directly depend on the continuous inputs X^c. In the context of state automata, this is called being of *Mealy* type. In contrast, *Moore* type automata define a simpler output function $\lambda^c : Q \to Y^c$ that does not directly depend on the inputs, but only on the state values. Their main difference lies in execution of coupled systems, which we will discuss below.

The remaining functions describe the discrete dynamics: δ_{ext} is executed for handling incoming (external) events, δ_{int} executes internal events, which are triggered when the time duration $ta(s)$ of the state s is exceeded. In other words, the non-negative real number $ta(s)$ specifies how long the atomic remains in the given state s in absence of incoming events. The function λ^d computes the (discrete) output values.

The function δ_{conf} is the so-called confluent[1] transition function and is the same as defined by Parallel DEVS (DEVS). It is executed whenever a hyPDEVS atomic receives an external event at the same time as an internal transition is scheduled (i.e. $e = ta(s)$) in order to resolve the collision and decide the next state at the Atomic level. The most common implementation (and in fact the one suggested by Zeigler as the default definition [320]) is

$$\delta_{\text{conf}}(s, e, x) = \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x), \tag{5.2}$$

specifying that concurrent events are resolved by first carrying out the internal event followed by the external event. However, in general hyPDEVS also allows other definitions

[1] The term confluent (as in "flowing together") reflects the intention that every execution path should always produce the same behavior in the end.

(that might even do something completely different) without reference to δ_{ext} and δ_{ext} to express special circumstances.

This aspect is different than with DEV&DESS, which, instead of δ_{conf}, defines a tie-breaking function *Select* at the Coupled level in order to resolve concurrent events by specifying an execution order of all the components within and allowing only one component to be activated at any time, thereby enforcing a globally serialized execution [320]. Avoiding this serialization imposed by the coupling presents a modeling advantage for the user in terms of improved modularity and thus reusability as the behavior of an Atomic should not depend in its coupling environment. In addition, this improved independence allows for hyPDEVS and PDEVS Atomics to be executed in parallel (hence the term Parallel DEVS).

This parallelization, however, makes it necessary to introduce *bags*[2] of input events (as opposed to sets). These bags can collect multiple input event, which may arrive during different iterations at the same point in time, thus recognizing that inputs may arrive in any order [130].

5.2.2 hyPDEVS Coupled

In addition to atomic hyPDEVS, the formalism also specifies coupled hyPDEVS models, which are comprised of an external input/output interface, sub-components (which must again be hyPDEVS components) and coupling relations. Coupled systems can be arranged hierarchically, meaning they can be incorporated just like an atomic into a larger coupled system. This property allows to construct modular hierarchical (tree-like) models in a component-based manner. Formally, a hyPDEVS coupled system is defined as

$$N_{\text{hyPDEVS}} = \left\langle X_N, Y_N, D, \{M_d\}_{d\in D}, \{I_d\}_{d\in D\cup\{N\}}, \{Z_d\}_{d\in D\cup\{N\}} \right\rangle, \tag{5.3}$$

where

$$\begin{aligned} &X_N = X_N^d \times X_N^c \text{ is the set of external inputs of the network,}\\ &Y_N = Y_N^d \times Y_N^c \text{ is the set of external outputs of the network,}\\ &D \text{ is the index set (i.e. set of component references),} \end{aligned}$$

for each $d \in D$

$$M_d \text{ is again a hyPDEVS model (i.e. sub-component of the coupling),}$$

and for each $d \in D \cup \{N\}$

$$I_d \text{ is the influencer set of } d,$$

[2] In contrast 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.

i.e. the index set of components that influence the component d via direct couplings, where $I_d \subseteq D \cup \{N\}$, $d \notin I_d$. The exclusion $d \notin I_d$ implies that no direct feedback loops are allowed, i.e. no output of a component may be connected to an input of the same component. Furthermore, for each $d \in D \cup \{N\}$, Z_d is the interface map for d specifying the coupling relationships, and is divided into (memoryless instantaneous) functions for discrete and continuous couplings $Z_d^d : \times_{i \in I_d} YX_i^d \to XY_d^d$ and $Z_d^c : \times_{i \in I_d} YX_i^c \to XY_d^c$ with

$$YX_i^d = \begin{cases} X_i^d & \text{if } i = N \\ Y_i^d & \text{if } i \neq N \end{cases}$$

$$XY_d^d = \begin{cases} Y_d^d & \text{if } d = N \\ X_d^d & \text{if } d \neq N \end{cases}$$

and likewise for YX_i^c and XY_d^c. In addition to separating couplings between discrete and continuous inputs/outputs, these sets basically distinguish the couplings between couplings within the network (output Y_i to input X_d), couplings to external outputs (Y_i to Y_N) and couplings from external inputs (X_N to X_d). Figure 5.2 illustrates these interface mappings graphically.

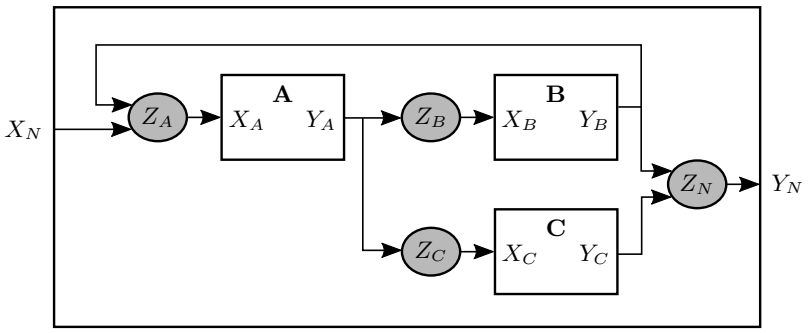


Figure 5.2: Coupled system specification using interface maps

This is the most general coupling specification, where the interface map Z_d specifies how the input values of component d are derived from the outputs (or external inputs of the network) of its influencers $i \in I_d$. This way, arbitrary couplings may be realized in theory.

Here, components are coupled exclusively through their input and output interfaces. Components do not have the possibility to influence the states of other components directly. All interactions have to be done via the interfaces. Discrete events generated by one component at its output are transmitted as messages along the couplings to the inputs of another component, where they cause external events and state transitions.

In order for the coupling to be well-defined, it is not allowed to contain *algebraic cycles* with zero delay, i.e. each feedback loop has to contain at least one component the output of which can be computed without knowledge of its input. Recall from above that hyPDEVS Atomics can be divided into Moore type and Mealy type. So, in order for

a coupling to be well-defined, in each cycle of output-to-input connections of coupled components there must be at least one component of Moore type [319]. Otherwise, the resulting algebraic loops would have to be solved simultaneously by searching for consistent values (e.g. using fixed-point iteration). While this approach is possible in principle and commonly adopted for Differential-Algebraic Equation (DAE) models [124], and if a unique solution exits then a consistent system can result, however, DEVS-based formalisms (in particular DEV&DESS and DTSS, see [319]) obviate the problem by requiring that no algebraic loops are present. In other words, the modeler has to take care that each feedback loop contains at least one Atomic of Moore type [130]. This ensures that the legitimacy[3] condition is satisfied, which will be discussed in the next section.

5.2.3 Legitimacy

There are some restrictions for modeling hyPDEVS components, most notably the *legitimacy condition* inherited from Classic DEVS. Legitimacy means that, in order for a model to be well-defined, only a finite number of events may occur in a finite amount of time, for every possible set of initial conditions. A more precise definition uses an extension of the internal transition function to its iterative form

$$\delta_{int}^{+} : S \times \mathbb{I}_0^+ \to S \tag{5.4}$$

over an index set $\mathbb{I}_0^+$ and defined recursively by

$$\delta_{int}^{+}(s, 0) = s, \tag{5.5}$$
$$\delta_{int}^{+}(s, n+1) = \delta_{int}(\delta_{int}^{+}(s, n)). \tag{5.6}$$

The value $\delta_{int}^{+}(s, n)$ is the state reached after n iterations, starting at state $s \in S$, without external events. The function $\sum(s, n)$ accumulates the time advances the system takes in the course of these n transitions:

$$\sum : S \times \mathbb{I}_0^+ \to \mathbb{R}_0^+ \tag{5.7}$$

defined recursively by

$$\sum(s, 0) = 0, \tag{5.8}$$
$$\sum(s, n) = \sum_{i=0}^{n-1} ta(\delta_{int}^{+}(s, i)). \tag{5.9}$$

Using these definitions, legitimacy can be formally defined. A DEVS system is legitimate if for each $s \in S$ it holds that [319, p. 158]

$$\lim_{n \to \infty} \sum(s, n) \to \infty. \tag{5.10}$$

[3] In particular, a potential divergence of a fixed-point iteration for solving the algebraic loop would result in infinite iterations and thus an illegitimate model.

In practice, this condition prohibits atomics going into an infinite loop of internal events without advancing time beyond a certain point. Legitimacy provides a necessary and sufficient condition for the system specified by DEVS to be well-defined. A DEVS system would not be legitimate for example if there was a cycle of state transitions that only contains transitory states, i.e. states for which $ta(s) = 0$, meaning hat, once entering this cycle, time would not advance anymore. Such a cycle may occur both at the Atomic level (by an invalid definition of δ_{int}) and at the Coupled level, by coupling several Atomics forming an endless loop.

For hyPDEVS, the legitimacy property for DEVS can be extended in the same way as it is done for DEV&DESS in [319, p. 232] to define the property of *state event legitimacy*. A hyPDEVS system is state-event-legitimate if and only if

$$\sum(q, \omega) < \infty \tag{5.11}$$

for all admissible inputs $\omega : (t_1, t_2] \to X$, where $\sum(q, \omega)$ is the number of state events, i.e. executions of the internal transition function in the time interval $(t_1, t_2]$ and q the initial state $q \in Q$ at time t_1.

The equivalent of this condition in the continuous domain is the *Lipschitz condition*, stating that

$$||f(q, x) - f(\tilde{q}, x)|| < k||q - \tilde{q}||_2 \tag{5.12}$$

for all pairs $q, \tilde{q} \in Q$ and input values $x \in X$, where f is the rate of change function, k is a constant and $|| \cdot ||_2$ is the Euclidean norm. This condition guaranties that a unique, well-defined state trajectory exists for every state and input [319, p. 164]. All these conditions must hold for a hyPDEVS system to be considered legitimate.

5.2.4 Closure under Coupling

All DEVS-based formalisms provide an important property, called *closure under coupling*. It guarantees that a well-defined coupling of systems in a DEVS formalism defines a basic (atomic) system in the same formalism [319]. In other words, a coupled system always behaves the same as an equivalent atomic when looked at from the outside. The implication is that closure under coupling allows to use networks of systems as components in a larger coupled system, thereby giving rise to the construction of models in a hierarchical, modular fashion [163]. For hyPDEVS, closure under coupling requires that the formalism provides a means to specify components with intermingled discrete and continuous expressions.

5.2.5 Simulation Execution

Executing the simulation by computing the state and output trajectories from the given model specification together with initial state values and time segments for all input ports is the task of a simulation algorithm. Hereby, the execution semantics are formally given

in terms of an abstract simulator algorithm. The common approach for DEVS simulation engines is to directly map the hierarchical structure of the model by distinguishing between *Simulator* and *Coordinator* classes. Figure 5.3 depicts this mapping of a hierarchical model to an hierarchical abstract simulator. One advantage of this approach is that it can be formally shown that this hierarchical structure of Simulators and Coordinators correctly simulates the model [319].

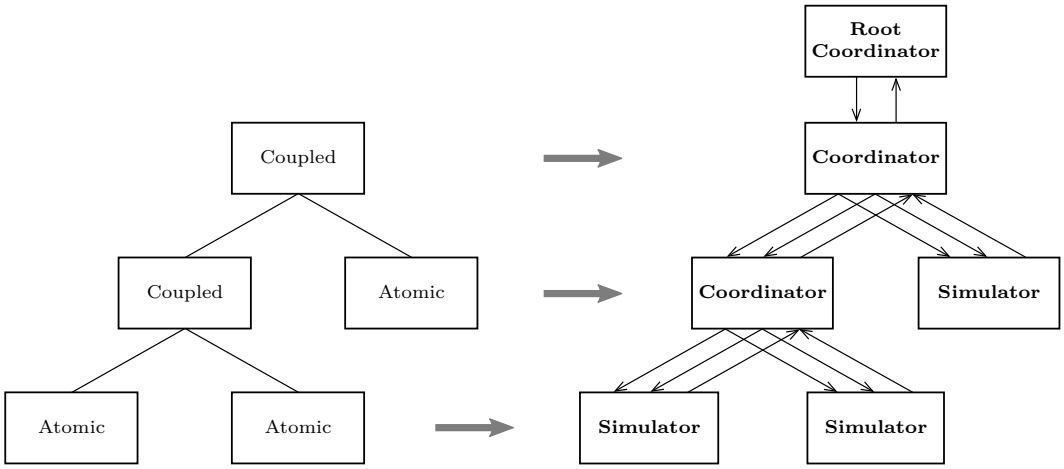


Figure 5.3: Hierarchical simulator (right) mapping a hierarchical hyPDEVS model structure (left) and employing message passing between its elements

A Simulator implements steps to call and execute atomic models, while a Coordinator, assigned to a coupled network, is responsible for the correct synchronization of its network elements and propagating output events. A Root Coordinator at the top of the hierarchy implements the overall simulation loop and is responsible for advancing the simulation time and initiating the simulation cycles. For communication, Simulator and Coordinator components realize a generic message protocol using different types of messages, including propagating inputs to children, invoking executions or receiving outputs. Although the abstract simulators typically employ message passing, they can be realized in different ways. In fact, there are other approaches that flatten the simulator hierarchy, thus avoiding additional message traffic and improving performance [163].

The abstract simulator for Atomic and Coupled hyPDEVS can be defined based on the Simulator and Coordinator for Parallel DEVS (PDEVS) and extending it with ODE handling with additional handling of state events similar to the Differential Equation System Specification (DESS) abstract simulator, see [319, p. 216]. While the simulation execution of discrete systems on a digital computer is relatively straightforward in that they natively employ a discretized time base, the simulation of continuous systems on a digital computer involves computing continuous-time behavior in discrete steps. This can be accomplished by means of numerical integration techniques according to

$$q(t_{i+1}) = \Phi(q(t_i), f(q(t_i)), x(t_i)), \tag{5.13}$$

where Φ describes the integration function depending on the particular method at hand. It uses the previous state value and input at t_i (or possibly multiple previous values) to estimate the state at the next time instant t_{i+1}. The state value can then directly be used to give an estimate for the output value by applying the output function

$$y(t_{i+1}) = \lambda^c(q(t_{i+1})). \tag{5.14}$$

More details on handling numerical integration in connection with DEVS will be discussed in the next section. The discrete and continuous parts are synchronized by an extended root coordinator. During simulation, they alternate in model execution. While the discrete part executes the state transitions at the event times, the other part computes the continuous state trajectories in between [319]. The operational semantics have been informally described above. More formal details are given in [76, 319].

5.2.6 Handling ODEs

The hyPDEVS formalism itself does not specify how exactly to handle numerical integration of the ODEs during simulation. In connection with DEVS, different approaches are possible, which we will briefly sketch out in the following.

In general, one can distinguish between explicit and implicit integration schemes [123]. While implicit schemes offer a wider stability, they typically require iterative computation where values have to be exchanged multiple times within one simulation cycle, which is why it is crucial in this case that communication must be implemented most efficiently. We will stick to explicit integration schemes, which have been proven to be sufficient for our application domain.

QSS Embedding: In contrast to conventional ODE solvers, which discretize the time domain and compute state values accordingly, solvers based on the QSS method [166] discretize the state space (with a fixed or variable quantum Q) and calculate the next point in time where state values have changed by a quantum Q. As a result, a discretized QSS model essentially becomes a Discrete-Event Simulation (DES). So, by including a QSS integrator (as an Atomic) into the model, everything can be embedded into DEVS (or PDEVS, respectively) and simulated using the standard simulation engines. However, special care has to be taken for the embedding to properly handle state events. This embedding approach has been described in [318] for embedding DEV&DESS into DEVS, and Deatcu et al. describe in [77] the same idea for embedding hyPDEVS into PDEVS.

The biggest advantage of this approach is that it seamlessly integrates with Discrete-Event Simulations and that native DEVS simulation engines can be employed for simulation without the need for hybrid extension. The drawback, on the other hand, is that, since this embedding is done at the coupling level with additional Atomics for QSS integration and state event handling, the compositional complexity of the overall model increases substantially and that the modeler has to take care of this embedding manually since the integrator becomes part of the model instead of separating model and simulator.

Wrapper Approach: A different approach is presented in [77]. The authors employ an ODE wrapper concept, where a closed representation of all continuous equations is derived from the hyPDEVS description automatically at runtime, which can then be computed by a single ODE solver algorithm alongside the discrete-event PDEVS engine. The runtime execution is orchestrated by a modified Root Coordinator that operates in three phases: initialization, discrete phase, and continuous phase, as described in [76]. Based on the minimum time stamp of the next internal event (in any component), the continuous cycle computes the differential equations until the next event becomes imminent. At that time, the Root Coordinator enters the discrete phase and executes all relevant Atomic components according to standard PDEVS behavior. If all internal and external transitions have been executed for that particular time step, the simulation time can be advanced, again by entering the continuous simulation first. The resulting coupling scheme between the continuous solver and the discrete-event engine is similar to the canonical synchronization model described by [109], where the continuous part is executed first until the next event. After that, the discrete events are processed and the cycle continues.

The advantage of this method is that the continuous part of the overall model can be computed by a single centralized solver, thereby simplifying the coupling scheme and reducing communication overhead. The central solver always knows beforehand when the next discrete event[4] occurs anywhere in the model and can stop accordingly [130]. In contrast to the stand-alone approach (see below), there is no need for discarding steps because they have been interrupted by unforeseen external events. Moreover, the ODE wrapper can employ advanced and established numerical methods for solving the ODEs that ar common in engineering applications (e.g. implicit integration schemes, step size control). It also avoids unnecessary model complexity by handling the numerical integration at the simulator level instead at the model level. And since it preserves the structural information and derives the wrapper only at runtime, reusability of hybrid Atomics is still attained. On the downside, the enforced centralization prevents the simulation to exploit parallel execution, which could improve runtime performance.

Stand-alone Approach: In order to still retain parallelization, a more advanced approach is to handle the ODEs decentralized for each individual Atomic and still at the simulator level. This requires a coupling scheme similar to distributed hybrid co-simulation with event-accurate synchronization and rollback possibilities [266, 17]. See Section 3.2.2 for additional details. Since the ODE computation is localized to the individual Atomic, it has to be able to react to unforeseen external events, by discarding the current step and repeating the computation.

The solver is being incorporated into the Atomic Simulator and handles communication with the discrete part of the Atomic independent of the surrounding DEVS message

[4] This includes external as well as internal events, since, globally, every external event is triggered by an internal event of another Atomic. Also, state events are not a problem in this case as they are detected by the ODE solver itself anyway.

passing infrastructure. It is therefore necessary to employ solver algorithms that are in principle independent from DEVS, i.e. standalone. Of course, all classic time-based ODE solvers can be used for this. Also, some advances have been made towards a stand-alone QSS solver implementation recently. The ODE solver can be implemented at the object level, as part of the simulator, in order to keep it separate from the model and keep the compositional complexity low.

The advantage of this method is that combines the benefits of classic ODE solvers as well as QSS methods with the ability of modular and decentralized computation while at the same time keeping the computational complexity low. However, this is also the most complex method to implement into a simulator and orchestrating the decentralized computation is not a trivial task and comes with communication overhead. Also, the Coordinator still needs to be extended to include continuous signals instead of using a pure discrete Coordinator as in the embedding approach.

This approach was also the one used for the stand-alone hyPDEVS simulator implementation, which is described in Section 5.6.

5.2.7 Remarks

The coupling specification in Equation (5.3) is similar to the one for the DEV&DESS formalism described in [319], with two notable exceptions: Equation (5.3) does not require the *Select* function for arbitrating concurrent events on the coupling level, and the interface mapping Z_d strictly separates discrete and continuous couplings. While the coupled DEV&DESS specification allows – with restrictions – to couple discrete and continuous interfaces together and therefore to build multi-formalism networks with components of different formalisms, we here restrict coupling relationships to the discrete and continuous domains, respectively. This is due to the restrictions that come with multi-formalism couplings which are grounded in the inherently different semantics between discrete and continuous signals and the non-trivial semantic alignment, most prominently the restriction in DEV&DESS couplings that continuous output signals are only allowed to be piecewise constant. For more details, we refer to [319, p. 239].

5.3 Cube Modeling

With the hyPDEVS formalism, we now have the tool at hand to develop hybrid discrete/continuous simulation components for the domain of industrial production systems. In Section 4.3, we have introduced Cubes as a unified modeling concept for component-based development of interdisciplinary application models. We want to briefly re-iterate on the simulation aspect of Cubes and how they are implemented using hyPDEVS.

Cubes provide a modularization to build simulation models in a hierarchical manner – Cubes encapsulate a well-defined behavior and can be used to build larger Cubes. This is in alignment with many common simulation tools that offer component-based modeling, such as Modelica [104, 283]. Even more so, these tools almost exclusively employ an

object-oriented software implementation of these components as reusable model classes that can be instantiated in different contexts. In the same way, Cubes are reusable building blocks to create new simulation models by combining predefined components. These components are grouped into a library of Cube classes [183].

Our premise was that these Cube models can be of hybrid nature at the component level, meaning not just that there can be discrete Cube and continuous Cube, but that even a single Cube might incorporate combined discrete and continuous behavior. This is one of the reasons why we chose hyPDEVS for implementation. However, Cubes can be seen as an abstraction of hyPDEVS in that a Cube may be comprised of not just one, but several hyPDEVS Atomics. The formalism makes no restrictions in this regard and offers the flexibility to implement hybrid components in whichever way is suited best for the individual model. In particular, the modeler is not forced to split the component along the boundary of discrete/continuous modeling with complicated and unintuitive interfaces. We will elaborate on this aspect in more detail in Section 5.5.3.

As mentioned in Chapter 4, for developing Cube models, instead of directly using the hyPDEVS description, it proved to be beneficial to first develop the models using a more abstract and high-level description and then later translate this description into a formal hyPDEVS specification. Since model development is usually a highly iterative process, it is faster not to immediately enforce formal accuracy, but instead to initially rely on a semi-formal description (e.g. by using graphical state diagrams) that is also more intuitive for domain experts and software engineers and easier to communicate to others.

Using this process of model development, we have developed and implemented a library of Cube components, of which we will provide an overview in the following. Then, we will take a closer look at a representative example of a Cube model, namely a conveyor oven.

5.3.1 Interfaces

What is most important to consider when developing model components that are supposed to be interoperable is that they have uniform interfaces with common semantics. In particular, we distinguish three different types of interfaces, as illustrated in Figure 5.4:

- Material: For exchanging entity objects. Different entities can be distinguished by its attributes (see below).
- Energy: For exchanging energy, in particular power values (unit: Watt).
- Information: For exchanging all other kinds of interaction, in particular control signals, energy demand and temperature values.

While material and information interfaces constitute discrete ports, energy interfaces are handled as continuous. Distinguishing material, energy and information flow is a recurring scheme in modeling of industrial production systems and can be found in several other relevant publications as well, see for example [34] and [280].

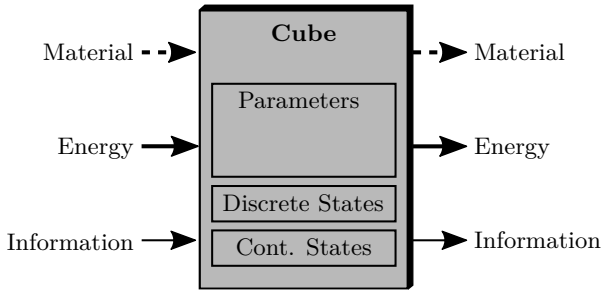


Figure 5.4: Generic Cube consisting of input and output interfaces for material, energy and information exchange. The internal behavior is comprised of discrete and continuous internal states and can be influenced by setting different parameters.

Besides their interfaces, Cube models also comprise an internal behavior characterized by internal states (discrete and continuous) and parameters.

5.3.2 Entity Modeling

Entities constitute the movable objects in the simulation that are exchanged between stations. They represent a workpiece that is being processed or a loaf of bread being baked. They can represent single workpieces as well as entire groups (e.g. a batch), depending on the desired resolution of the simulation. They could theoretically also represent other types of movable production resources, such as a pallet for carrying workpieces.

Entities are implemented as object classes and comprise different attributes that characterize its properties, like mass, product type, etc. Table 5.1 defines the main entity attributes used in the Cube library.

Table 5.1: Defined entity attributes

Name	Attribute	Unit
Identifier	ID	
Type	type	
Mass	m	kg
Temperature	T	°C
Heat capacity	c_p	J/(kg·K)
Job number	job	
Best before date	BBD	
List of sub-entities	ent	

Some of the attributes (e.g. product type) are necessary for logistic reasons, while others are required by the thermal computations. The entity can carry a list of sub-entities that

allows to define groups of entities that can be batched together and split up again. In addition, different weight factors have been defined as entity attributes that are relevant e.g. to calculate the Carbon Footprint of Products (CFP). For more details on this, we refer to [256].

The values of the attributes may change during the simulation, for example the temperature when the entity enters an oven or the mass of the workpiece after a machining operation. The values are set by the stations.

5.4 Cube Library

Figure 5.5 gives an overview of the most important Cube classes. The classes are divided into four main categories, depending on which industrial domain they represent:

- Production equipment (blue): for processing entities.
- Logistics components (purple): for transporting, handling and storing entities.
- Energy system (red): for supplying the production with final energy, including energy conversion, distribution and storage.
- Building components (green): for thermal building modeling, including thermal zones and heat transfer.

5.5 Example: Conveyor Oven

In the following, we briefly present the model of an Oven Cubes, in particular a conveyor oven. This model can be considered to be representative for all production and logistics Cubes, which is why we want to study it in more detail. It shows the full potential of integrated hybrid modeling with hyPDEVS and includes all important aspects of a Cubes model, in particular discrete and continuous behavior on the component level (with differential equations, a non-trivial state machine, discrete entities, time-driven internal and external events) as well as their interactions in terms of state events. It also includes the basic mechanism for moving entities within the station as well as between stations, which also shows up in other Cubes.

5.5.1 Cube Model

The Oven Cubes, shown in Figure 5.6, represents a generic station for the thermal treatment of goods, both for heating and cooling [130]. This includes, for example, an oven or freezer for baked goods, or a furnace for hardening steel workpieces. This station is designed as a conveyor belt[5] with given capacity N and holding time t_B. It accepts

[5] For $N = 1$, the model is also able to represent batch behavior. However, batch groups would need to be merged beforehand using a Combiner Cubes.

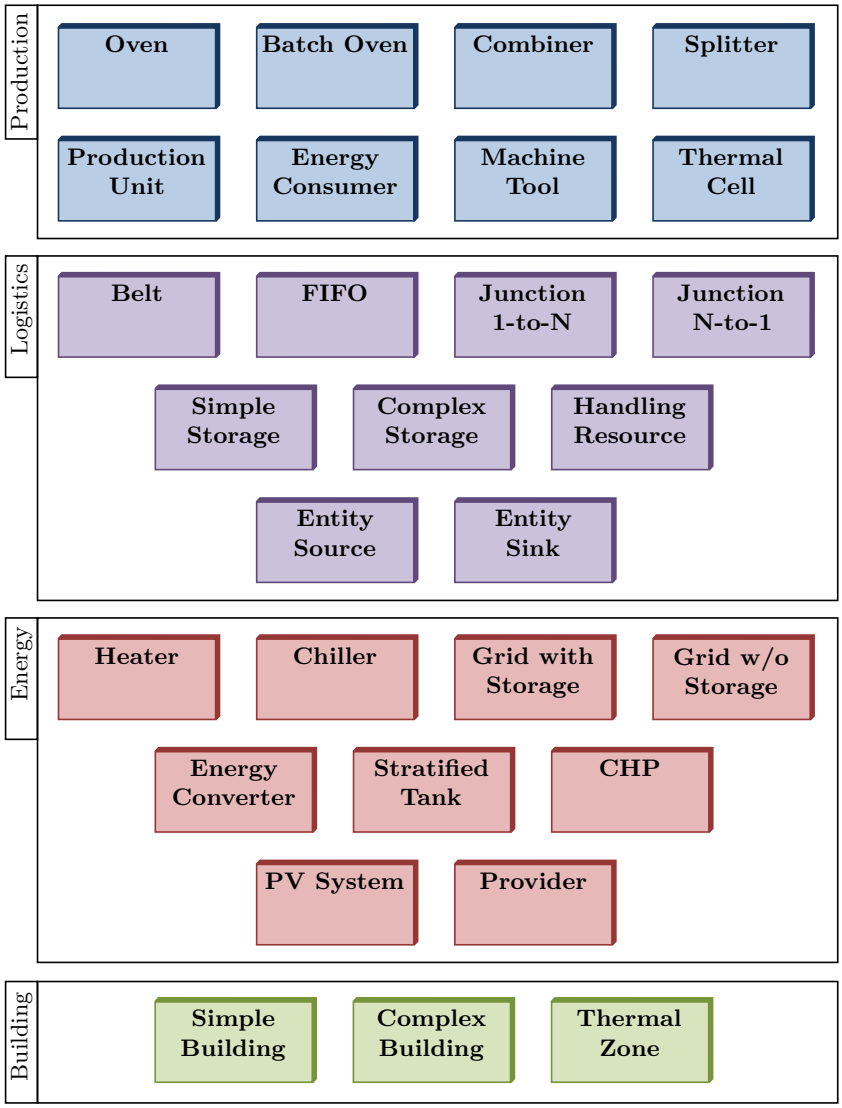


Figure 5.5: Library of Cubes models, including Production Cubes (blue), Logistics Cubes (purple), Energy Cubes (red) and Building Cubes (green)

entities at the input E_{in}, holds them for the duration t_B while moving them successively along the conveying distance and then outputs them again at the port E_{out}. The Oven takes a *Pplan* parameter, which determines necessary[6]. setup times for pre-heating or pre-cooling the station. It also allows to change process parameters depending on the product type.

Figure 5.7 depicts the *discrete* internal behavior of the Oven model, governing the material flow and information exchange, represented semi-formally as a state diagram.

[6]In contrast to most other stations, the *Pplan* parameter is mandatory for the Oven Cubes.

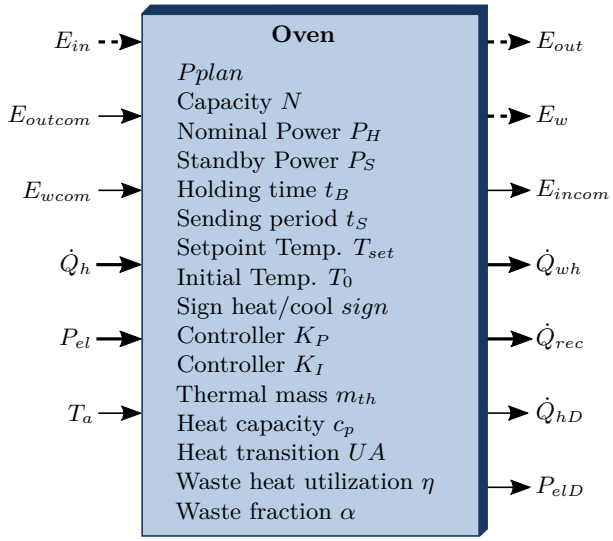


Figure 5.6: Oven Cubes with interfaces and internal parameters

The initial state is `standby`, from where the *Pplan* may switch the state to `off` or start setup (state `heating`). When the requested setpoint temperature T_{set} is reached, the Oven switches to `waiting`, after which it is ready to accept entities, while maintaining the temperature. Incoming entities at the input E_{in} are confirmed at E_{incom} via an acknowledgment signal and then stored in a list *ent* (state `incoming`), after which the state switches to `holding`. Here, further entities may enter (via state `incoming`). After the time interval t_B/N, the entities are shifted periodically by one place (state `update`) and the farthest entity is output at the end (state `output`), while also updating the temperature attribute of the outgoing entity. Optionally, a second entity may be split off (with mass fraction $\alpha \cdot E.m$) that represents waste[7] and which exits at port E_w. In case the outgoing entities are not accepted by the subsequent downstream stations, the Oven keeps waiting for the E_{outcom} and E_{wcom} events and continuously tries to resend the entities (interval t_S). After the entities have been acknowledged, the entity list is updated and the Oven continues in the state `holding` or `waiting`, depending on whether entities are remaining or not. From the state `waiting`, the Oven can be turned off, again via a *Pplan* signal.

All states marked with «`transitory`» are transitory states, meaning $ta(s) = 0$, which, upon entering, are immediately left again (outgoing transition is `true`). These states are special in that they are not mandatory and may be avoided. In the state diagram, these states could be replaced by corresponding trigger actions, i.e. actions associated with the corresponding transition rather than a separate state. However, the variant with transitory states was chosen because of its clarity and improved readability. In the hyPDEVS implementation, the transitory states do not need to be implemented

[7] This can make sense for example in a waffle oven where parts of the dough may fall off and exit through a collecting bin.

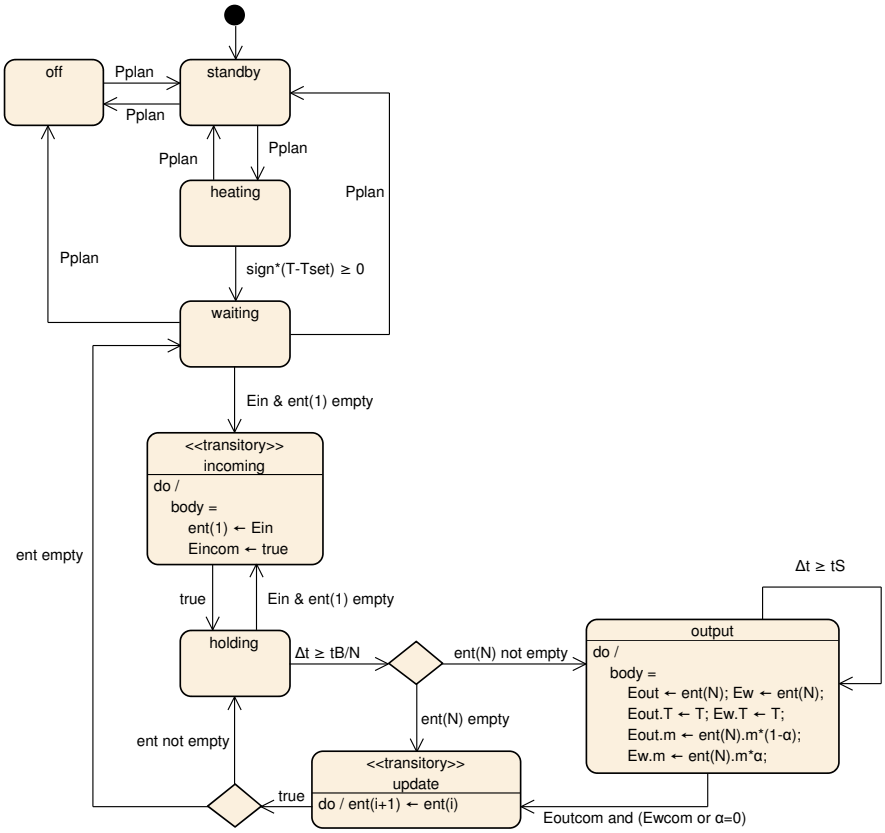


Figure 5.7: State diagram describing the discrete behavior of the Oven Cubes

explicitly, and instead may constitute actions performed *within* the transition functions (δ_{ext}, δ_{int}, δ_{conf} or δ_{state}) on the way to entering the subsequent state. This will become clearer in Algorithm 5.1.

The *continuous* behavior of the Oven is modeled using energy balance equations. In particular, the interior temperature T changes depending on incoming power $\dot{Q}_h$, outgoing heat transfer $(T - T_a) \cdot UA$ and overall thermal mass:

$$\frac{dT}{dt} = \frac{\dot{Q}_h - (T - T_a) \cdot UA}{m_{th} \cdot c_p + \sum_{E \in ent} E.m \cdot E.c_p}, \tag{5.15}$$

with initial condition $T(0) = T_0$. The input T_a denotes the ambient temperature, c_p the specific heat capacity, m_{th} the thermal storage capacity (mass), and UA the heat transition coefficient[8] through the Oven walls. The term $\sum_{E \in ent} E.m \cdot E.c_p$ denotes the sum of the heat capacities of all entities $E \in ent$ inside the Oven.

[8]The parameter UA is typically given as $UA = U \cdot A$ with the specific heat transition coefficient U and the surface area A.

The heat transfer manifests as waste heat, which is emitted to the environment (output Q_{wh}), together with the converted electric power P_{el}. Hereby, a fraction η of the overall waste heat may be recoverable and is output separately (output Q_{rec}) according to

$$\dot{Q}_{wh} = ((T - T_a) \cdot UA + P_{el}) \cdot (1 - \eta), \tag{5.16}$$

$$\dot{Q}_{rec} = ((T - T_a) \cdot UA + P_{el}) \cdot \eta. \tag{5.17}$$

During operation, a *controller* is responsible for computing the request for thermal energy (output $\dot{Q}_{hD}$), which is to be supplied (by the energy system infrastructure) at the input $\dot{Q}_h$. Thereby, the controller implements a PI control strategy, which is additionally saturated with the nominal power P_H according to

$$Q_{hD}(t) = \min\left(P_H, K_P \cdot (T(t) - T_{set}) + K_I \cdot \int_0^t (T(\tau) - T_{set})\, d\tau\right) \tag{5.18}$$

with controller parameters K_P and K_I. In addition, the Oven also requests electric energy according to

$$P_{elD} = P_S, \tag{5.19}$$

except for when it is turned off (state `off`).

5.5.2 Remarks

Alternative Conveyor: An alternative way of modeling the conveyor part of the Oven would have been to model a continuous conveyor instead of discrete storage bins, on which each entities are transported independently of each other by scheduling a corresponding outgoing event at t_B later after an arriving entity. Some tools for material flow simulation do use this variant because the simulation uses fewer events and therefore offers better performance. However, in this case, in order to offer realistic behavior, the dimensions of the entities would need to be taken into account (in addition to the capacity) to detect any collisions on the conveyor belt. If this were not done, entities would be able to be transported "on top of each other" on the belt, which would be unrealistic in many cases and distort the material flow. This is why we opted for the simpler approach with discrete bins.

Simplifications: The model follows a few simplifications. For one thing, while the entities have a heat capacity and can therefore store heat, the model neglects the transient heat exchange effects between Oven and entities. When a new entity enters the Oven, it immediately assumes its internal temperature. However, the model still respects the overall energy balance.

Inverted Operation: The user-defined parameter *sign* allows to invert the thermal behavior of the Oven Cubes to operate it as a Cooler or Freezer, in which case (i.e

sign=-1) the incoming heat energy Q_h (as well as the demand output Q_{hD}) must also have a negative sign. This is a global convention that spans across all Cubes models and allows for consistent energy balances without the need to distinguish different ports for heating and cooling.

Power Supply: The fact that the power supply is only considered as abstract energy flow (as opposed to e.g. gas flow with internal combustion) allows to cover electrically powered ovens, those fed by district heat as well as gas ovens (by modeling the energy flow that is carried by the gas) and others.

Event Types: When looking again at the state diagram in Figure 5.7, all three types of event transitions, which are allowed by hyPDEVS are present:

- Time-driven (internal) events: Processing finished ($\Delta t \geq t_B/N$),
- External events: Incoming E_{in}, E_{outcom}, etc.,
- State events: Reaching internal temperature ($sign \cdot (T - T_{set}) \geq 0$).

5.5.3 hyPDEVS Model

The semi-formal model description of the Oven Cubes in terms of state diagram and continuous equations has to be translated into a hyPDEVS-compliant model by providing definitions for the functions specified by hyPDEVS in Equation (5.1) and Equation (5.3) [130]. Unfortunately, this is not a trivial process and several hyPDEVS-related aspects have to be taken into account.

From an implementation perspective, it might be a good idea to split into several hyPDEVS Atomics. This could make it easier for the simulation and software engineer to manage complex Cubes models, because, on the one hand, this aids in further separation of concerns by having certain atomics be responsible for certain encapsulated functionalities (e.g. temperature controller, reading the *Pplan*) where combining them into a single Atomic would make the model less comprehensible and more error-prone. On the other hand, it enables reuse of implementation artifacts across Cubes models. It seems intuitive to implement the Atomics in an object-oriented manner as Classes and then instantiate them for reoccurring use in different Cubes. However, multiple Atomics add compositional complexity due to the higher number of components and connections, which, in many cases, can be hidden from the application engineer by masking them within Cubes.

For the Oven Cubes, we decided to split the implementation into three Atomics, illustrated in Figure 5.8. Besides the *OvenAtomic* incorporating the main behavior, managing the *Pplan* table is outsourced into a *PplanSource* and the temperature controller is implemented in a *Controller* Atomic. All three are encapsulated within a hyPDEVS Coupled so as to keep the structural and interface definition of the overall Cubes intact.

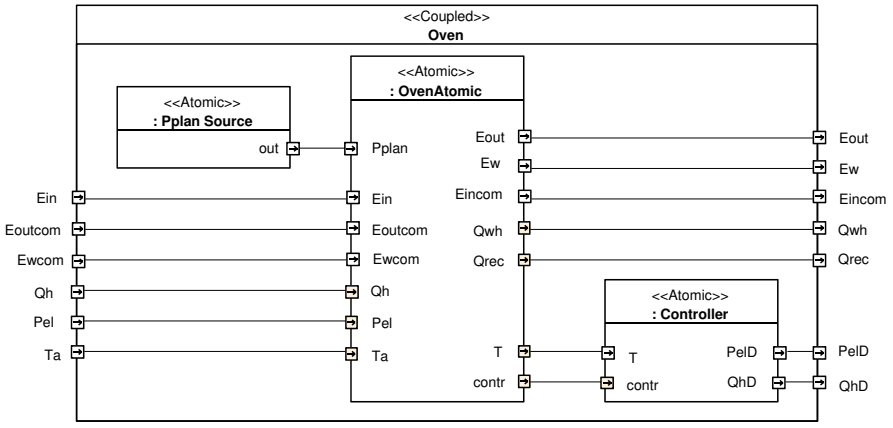


Figure 5.8: Implementation of the Oven Cubes as hyPDEVS Coupled, consisting of three Atomics.

The graphical representation in Figure 5.8 follows a SysML (internal block diagram) notation. It seems appropriate as it aims at, and is employed by, software engineers during the design and engineering of software systems for communicating internal details regarding behavior and implementation [103].

Realizing the controller as a separate Atomic has the advantage that various other controller strategies my be implemented in the same way without affecting the *OvenAtomic*. But instead of defining it as a separate Cubes, it makes more sense to encapsulate the controller inside of an existing Cubes. This way, the application engineers do not have to deal with the controller and its circuitry and therefore this aspect can be hidden from them. The reusability of the controller on the implementation level is still given.

The Algorithm 5.1 provides the hyPDEVS implementation of the *OvenAtomic* component as pseudocode. Some details have been omitted for reasons of brevity, the full pseudocode as well as the implementations for the *PplanSource* and *Controller* Atomics are given in Appendix A. For a complete specification, the code lists definitions for the functions ta, δ_{conf}, f, c_{se}, δ_{state}, λ^c, δ_{ext}, δ_{int} and λ^d (see Equation (5.1)).

Instead of a strictly mathematical notation for the hyPDEVS functions, we opt for a procedural representation because it is easier to read for complex models and easier to transfer into actual source code.

As it is common for DEVS-based models, the implementation introduces a state variable σ, which stores the time advance value (see the definition of ta in Line 1). In other words, it is the time remaining in the current state. The definition for δ_{conf} (Line 2) follows the convention of Equation (5.2). The function f implements the differential equation in Equation (5.15), while the algebraic equations (see Equation (5.16)) are calculated directly in the output function λ^c in a straightforward manner. The functions c_{se} and

Algorithm 5.1: hyPDEVS implementation of the *OvenAtomic* component

$ta(s) \leftarrow s.\sigma$;
$\delta_{\text{conf}}(s, e, x) \leftarrow \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x)$;
$f(s, e, x) \leftarrow \left(x.\dot{Q}_h - (s.T - x.T_a) \cdot UA\right) / \left(c_p \cdot m_{th} + \sum_i s.ent(i).m \cdot s.ent(i).c_p\right)$;
$c_{se}(s) \leftarrow sign \cdot (s.T - T_{set})$;
function $\delta_{\text{state}}(s, e, x)$**:**
    switch from state `heating` to `waiting`;
    return s;
function $\lambda^c(s, e, x)$**:**
    $y.\dot{Q}_{wh} \leftarrow ((s.T - x.T_a) \cdot UA + x.P_{el}) \cdot (1 - \eta)$;
    $y.\dot{Q}_{rec} \leftarrow ((s.T - x.T_a) \cdot UA + x.P_{el}) \cdot \eta$;
    $y.T \leftarrow s.T$; // output current temperature for controller
    return y;
function $\delta_{\text{ext}}(s, e, x)$**:**
    handle incoming Pplan signal and switch to the corresponding state;
    $s.\sigma \leftarrow s.\sigma - e$;
    if $x.E_{outcom}$ **and** *(*$x.E_{wcom}$ **or** $\alpha = 0$*)* **then**
        $s.ent \leftarrow shift(s.ent)$; // transitory state `update`
        switch to state `waiting` or `holding`;
    end
    if $x.E_{in}$ **and** $s.ent(1) = \emptyset$ **and** $s.state \in$ {`waiting`,`holding`} **then**
        $s.ent(1) \leftarrow x.E_{in}$; // transitory state `incoming`
        switch to state `holding` and schedule sending E_{incom};
    end
    return s;
function $\delta_{\text{int}}(s)$**:**
    if $s.state =$ `holding` **then**
        either switch to state `output` or shift and continue;
    else
        schedule resending entity or go passive;
    end
    return s;
function $\lambda^d(s)$**:**
    set $y.E_{incom}$ if scheduled;
    if $s.state =$ `output` **then**
        send E_{out} and E_w as long as no acknowledgment is received;
    end
    [...];
    return y;

δ_{state} are responsible for the single state event transition in the model between the states `heating` and `waiting` that triggers when the desired temperature has been reached, described by means of a zero-crossing function, i.e. $sign \cdot (T - T_{set}) \geq 0$. In Line 13, δ_{ext} is responsible for handling incoming events ($Pplan$, E_{in}, E_{outcom} and E_{wcom}) and λ^d generates all output events.

It is interesting to note the interplay of λ^d and δ_{int}, which are mostly executed in tandem. For example, after receiving an entity (Line 20), λ^c is triggered for sending E_{incom} (Line 33), followed by the execution of δ_{int} to switch the state and schedule the next event. The user has to keep this behavior in mind when implementing a model.

One exception to this λ^d-δ_{int} combination may occur during concurrent events when δ_{conf} is executed immediately after λ^d. Since this violation of the λ^d-δ_{int} combination might lead to some unintended behavior if not considered carefully enough, it is best to stick to the default convention of $\delta_{\text{conf}} = \delta_{\text{ext}} \circ \delta_{\text{int}}$ as this ensures that also in the concurrent case, δ_{int} is executed immediately after λ^d, thus respecting the λ^d-δ_{int} combination [130].

As mentioned in Section 5.5.1, the transitory states `incoming` and `update` are not modeled explicitly, but instead the corresponding actions are incorporated as part of the transition functions. They show up in Lines 17, 21 and 27.

Simulation experiments and results for the implemented Oven model are presented as part of the case studies, see Section 5.7. The Oven should serve as a representative example of a Cubes. For descriptions on the other Cubes models that are part of the Cubes library (see Figure 5.5), we refer to [29, 259, 258, 197]. Most aspects of these Cubes are modeled in a similar manner than the Oven, while others are much more straightforward, like the (purely continuous) energy Cubes.

5.5.4 Remarks

In the following, we want to briefly address some issues related to the hyPDEVS implementation [130], which apply not just to the Oven, but involve general considerations for all Cubes.

Alternative Implementations: It would also be possible to further split up the *OvenAtomic* into two Atomics, one implementing a Conveyor Belt handling the entities (discrete behavior) and a Thermal Cell for the thermal-physical (i.e. continuous) behavior, giving four Atomics for the Oven in total. In fact, this variant has been used for the stand-alone C++ simulator implementation, see Section 5.6. While this might come with some advantages regarding reusability, in particular reuse of the Conveyor Belt Atomic as a separate Cubes, the downside is the need for tight communication between the Conveyor and the Thermal Cell and more complex input/output ports, resulting in a rather awkward interface (e.g. exchanging entity mass and other attributes). It would also create more communication overhead and decrease simulation performance. In this case, the combined Belt and Cell Atomic seems to be the more harmonic choice. Reuse of the Belt functionality can instead be realized at the software level (as object classes).

Another disadvantage of separating the model into different Atomics is that individual parameters may potentially appear in multiple Atomics, thus creating redundancies. In the case of the Oven model, this concerns in particular the parameter T_{set}, which is needed in the *OvenAtomic* (for the state event transition) as well as in the *Controller* Atomic. One way to avoid this could be to provide an additional input/output interface for T_{set}. A different variant, which couples this parameter redundancy on a higher level, is described in Chapter 7.

Another implementation detail to note is that, while it would be possible to model the Oven Cubes as a single hyPDEVS Atomic and split that up at the object level (i.e. implement it in different classes), this approach would loose its formal rigor and thus the reliability and execution traceability that hyPDEVS provides. In fact, it would bypass the formalism and reduce the model implementation to a straightforward object-oriented paradigm.

Entity Push Semantics: The way entities can be exchanged between hyPDEVS Atomics requires some precautions. Consider for example the situation depicted in Figure 5.9 with two stations A and B trying to hand over an entity, call it E. The hyPDEVS specification only takes into account rudimentary event handling, i.e. Station A sends the entity E in form of an event and Station B has to process this event. However, if, for example, Station B is not able to accept E (e.g. because the storage capacity is reached), then it has to reject an incoming entity event. In order for E not to be dropped and lost, it is necessary that Station B notifies station A about whether or not E has been accepted[9] via an acknowledgment signal so that station A can keep E and resend it at a later time. An alternative to this *push semantics* would be to employ *pull semantics* by having station B always *request* an entity from station A before it is being sent. In any case, there has to be some form of additional communication between the two stations using an information channel. In the tradition of common material flow simulation tools and due to its simplicity, we opted for the push semantics as the primary principle for entity flow (although there are a few exceptions, e.g. fetching entities from a storage). In addition to the recipient having to acknowledge each incoming entity, the sending station has to keep resending a rejected entity periodically in order to avoid a deadlock. This is a drawback of this mechanism because of significantly more events potentially being generated. A more sophisticated approach might combine push and pull principles (e.g. start with push, then switch to pull after initial rejection), however, this would increase the model complexity.

Implicit Event Prioritization: As mentioned earlier, hyPDEVS employs a confluent transition function δ_{conf} instead of a *Select* function that would provide an explicit prioritization for executing Atomics, thus opening the door for parallel processing. Unfortunately, this also brings about some modeling challenges as no assumptions can

[9] While it would also be possible to notify the *rejection* of an entity, in terms of model robustness it is preferable to notify *acceptance*.

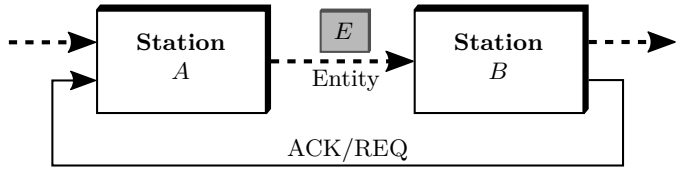


Figure 5.9: Simple example of two stations exchanging an entity, either using an acknowledgment signal (ACK) or a request signal (REQ).

be made anymore about input events that occur at a particular time step also arriving concurrently, i.e. during the same iteration of δ_{ext} [130]. This problem has also been described in [222] and has to be considered when implementing δ_{ext}. In addition, incoming events (that have arrived concurrently) are still being processed sequentially in the body of δ_{ext}, thus imposing an implicit prioritization of inputs, depending on which is being processed first. In the worst case, this might lead to some unintended behavior and the modeler has to be very careful when specifying δ_{ext}. As a particular example, δ_{ext} of the Oven Cubes processes an E_{outcom} event before an E_{in} event (see Lines 16 and 20 in Algorithm 5.1) because E_{outcom} needs to be checked first in order to potentially free up capacity before accepting another entity (in the same iteration).

Input Buffer: The described communication for exchanging entities also causes an additional iteration with message exchange during the same time step. Since typical DEVS simulation engines reset all external messages in between such iterations (since it is assumed that the events have already been processed), even if they happen at the same time step. For practical applications, however, it is useful to buffer incoming messages in between iterations 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 happens during entity exchange. Consider again the example in Figure 5.9 and assume that another entity E_2 arrives at the input of Station A at the same time it tries to send its entity E to Station B. Initially, Station A is not able to process E_2 immediately as long it is still blocked by E and it has to wait for acknowledgment from Station B. As long as the acknowledgment from B arrives in the same time step, station A might sill be able to accept E_2. But in order to achieve this, E_2 needs to be held (i.e. buffered) at the input until the entire exchange of E is finished, otherwise the event reset would delete E_2 from the input [130].

While such a buffering mechanism can be included as part of the model implementation, which is exactly what we have done within the Cubes library, it would be more effective to implement it as part of the communication mechanism at the object level and to abstract this communication away from the model level.

5.6 Implementation

The implementation of the simulation engine and Cubes model library was carried out in two stages: In the first stage, a first proof of concept implementation was developed to demonstrate the feasibility of the Cubes approach as well as DEVS-based hybrid simulation. The second stage then consisted of a stand-alone simulator re-implementation to be deployed into the field.

5.6.1 First Prototype

The first prototype was intended to be a proof of concept and to gain first-hand experience of applying DEVS-based hybrid simulation into practice in an industrial context. The MatlabDEVS Toolbox [76, 77] was used for this, which provides a simulation engine[10] for the hyPDEVS formalism. It is based on MATLAB [278] and employs the ODE wrapper approach for handling the continuous model as described in Section 5.2.6.

The Cubes models are created in an object-oriented manner by implementing classes of hyPDEVS Atomics, for which the common functions (δ_{ext}, δ_{int}, λ^c, etc.) have to be provided. These classes can then be instantiated and parametrized to create Coupled application models.

The proof of concept implementation includes essential Cubes models and a first case study (see Section 5.7). This implementation is described in more detail in [130]. The case study simulation was then also used to test and evaluate different simulation-based optimization strategies on top. More details on this are given in [261].

While the MatlabDEVS toolbox provided a practicable environment to carry out simulation experiments, it was not suitable to be used in productive operation in the field. Apart from minor implementation issues regarding Mealy type Atomics (see [130, 222]) and the fact that the simulation engine is not optimized for runtime, mainly the necessary (and ongoing) licensing costs for MATLAB prevented a practical deployment.

5.6.2 Stand-alone Simulator

After the first experiences from the proof of concept, a stand-alone simulator was developed in C++. This work was carried within the research projects BaMa and Adaptive Smoothed Production (ASPeCT) together with a project partner who took over the implementation.

For the stand-alone simulator, the first step was to implement a hyPDEVS simulation engine from scratch. The engine employs the novel standalone solver approach for decentralized ODE handling described in Section 5.2.6). Two different numerical integration techniques were implemented for this purpose, which can be selected individually for each

[10] In fact, the MatlabDEVS Toolbox also provides an extension for variable structure systems based on the Dynamic Structure DEVS (DEVS) specification [22]. But since this extension is not relevant for our purposes, we leave this aspect aside and focus on the hybrid simulation capabilities.

Cubes: The first is a classic Runge-Kutta RK45 method [226] with step size control and state event detection. The other is based on an advanced QSS method with logarithmic quantization [165]. In order to increase the simulation performance, an additional step was added during runtime translation to flatten the simulator hierarchy as described in [163] (cf. Figure 5.3) and to decrease the message traffic overhead.

In the second step, the library of Cubes models was also re-implemented in C++ using an object-oriented paradigm. Special focus was put on a reusable implementation, consistent interfaces and verbose logging capabilities for testing and debugging. These Cubes models could be verified against the MatlabDEVS implementation, for which a number of testing scenarios were created. This library was later extended with additional Cubes classes which were used in the larger case studies. A special user interface allows *Pplan*, *Aplan* and other simulation parameters to be changed to simulate different production scenarios.

The third step was to implement different case study applications using the Cubes classes, starting with the simple production line from the MatlabDEVS implementation with the intention to also verify the case study results against the independent implementation.

This stand-alone simulation tool then served as the basis to develop additional more advanced case study simulations, including an advanced production line (see also Section 5.8) [136], an entire bakery [262] and a production line for wafers [29], and to deploy the simulations as part of a simulation-based decision support tool for energy-aware production planning.

In the following, we want do present different case studies that demonstrate the application of the Cubes models. The case studies feature dynamic simulation of industrial production involving thermal processes and their interaction with Technical Building Services (TBS). They showcase the domain of industrial flow shop production, however, the simulation method should be applicable to other areas as well.

5.7 Case Study 1: Simple Production Line

5.7.1 Model Description

The case study is a simplified model of a real production plant of an industrial bakery that produces baked goods [229]. The conceptual model of this case study is depicted in Figure 5.10. It features a typical production line with machines, storage and conveyor belts, an energy supply system with heater and cooler, and a building model with thermal zones.

This simple case study was devised as a first proof of concept to showcase all important aspects of our production systems domain while omitting unnecessary details that would only add to the problem complexity. In particular, the case study includes

- cross-domain modeling with dynamic interdependencies,
- intertwined continuous and discrete dynamics, and

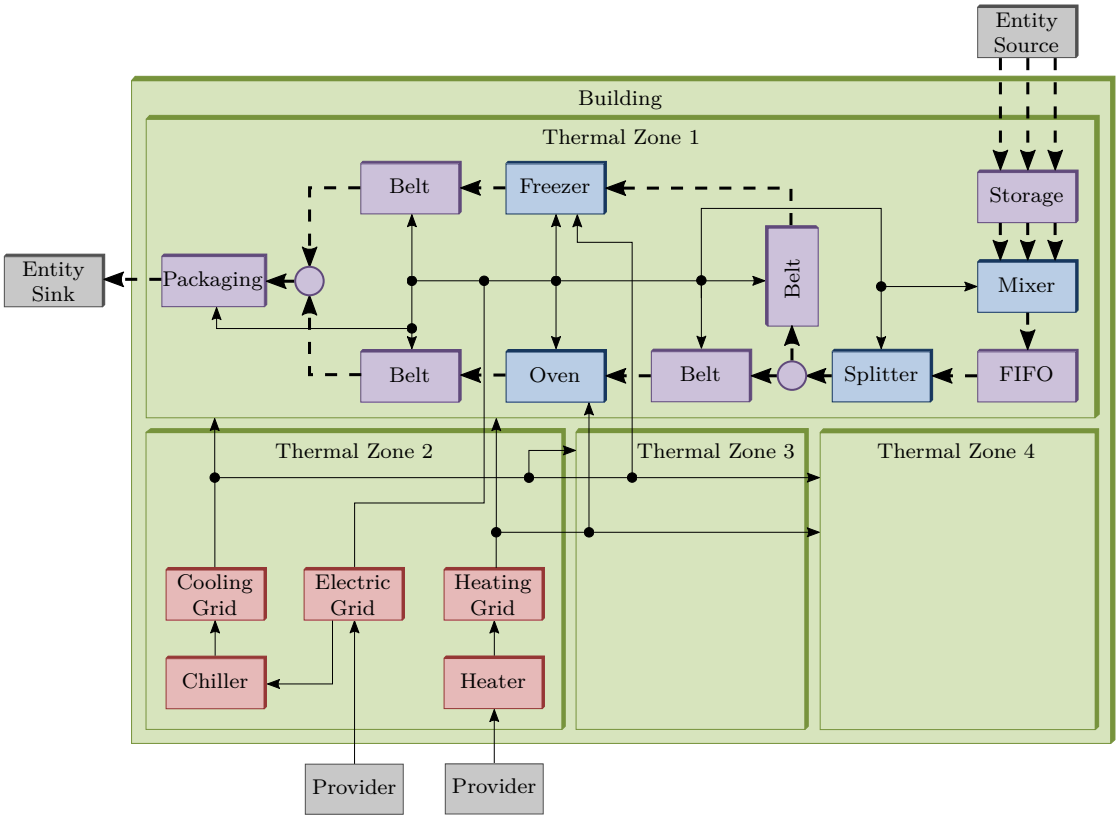


Figure 5.10: Conceptual model of Case Study 1: Simple Production Line. It includes production machines (blue), logistics components (purple), energy supply components (red) and thermal building zones (green).

- complex product flow with splitting, merging and batching.

The production and logistics components form a production line for two product variants: baked and frozen. Baked products pass an oven for baking while frozen products are frozen directly (in a freezer) without being baked. Both are designed as conveyor belts, meaning that new entities continuously enter the stations and leave on the other side. Since these products share all other stations, only one type of product can be produced at any time. For both products, respective ingredients are being pulled from the storage, after which they are mixed into a dough, divided into portions (splitting) and continue on different conveyor belts. After baking/freezing, the finished products are packaged in different quantities.

The building is modeled as a simple thermal compartment model with four thermal zones, each representing a distinct part of the facility: production hall, plant rooms and cold storage. These zones all have independent conditioning (for example, the cold storage is kept at 4 °C) and exchange thermal energy with one another according to the defined

wall topology. They also exchange thermal energy with the environment, for which a variable ambient temperature may be specified.

The energy system provides necessary technical building services for supplying final energy for the production machines as well as for heating and cooling the thermal building zones. The energy system in turn obtains its energy input from external providers for gas and electricity. It is comprised of a heater (powered by natural gas) that supplies heat to the oven and building, a chiller (powered by electricity) that supplies cold to the freezer and building, as well as respective energy grids responsible for distributing the energy. The heating and cooling grids also include thermal energy storage that models the storage capacity of the grid. All other production stations receive electric energy from the grid.

The production orders are executed according to a production schedule (*Pplan*), which is the main input vector to specify the production scenario. In addition to the order starting times, the *Pplan* also specifies the start of the setup processes for the oven and freezer. During these setup processes, the oven (or freezer) is preheated (or cooled) to the defined operating temperature before the products arrive at the station. This also implies that, in contrast to traditional discrete-event material flow simulations, the setup time is not a fixed parameter, but may change dynamically during the optimization process.

The case study further includes a work plan (*Aplan*) that specified the production steps and process parameters that depend on the product type, such as temperature set points, baking time or batching size. It serves as a look-up table for the individual stations and the process parameters may be different for different product types.

Although the production is kept simple and the number of products is limited, this does not undermine the goal of demonstrating the feasibility of the simulation method.

In the following, we present and compare exemplary simulation results in different scenarios in order to demonstrate the application of the case study model.

5.7.2 Scenario 1

The scenarios are mainly defined by the *Pplans*, which specify when which product is to be produced in which quantity. Table 5.2 shows the *Pplans* for two scenarios, which take place over one day (00:00 to 24:00 h). Both scenarios produce the same entities, but at different times. They are intended to provide a comparison of different production times under the same conditions (i.e. number of entities to be produced) [130].

All stations follow the same basic production schedule in order to enable frictionless entity flow through the production line. The *Pplans* for Oven and Freezer factor in an additional setup time for pre-heating and pre-cooling, respectively. The Storage at the beginning of the production line starts earlier in order to fetch and prepare the ingredients for production.

In addition to the *Pplans*, the simulation reads a work sheet (*Aplan*) that specifies process parameters, such as baking temperature, processing time or packaging quantity, as well

as external input data for ambient temperature conditions. These remain unchanged between scenarios.

Table 5.2: Production schedules for Scenarios 1a and 1b

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

Figure 5.11 and Figure 5.12 show the resulting entity flow in terms of numbers of entities in the stations over time for Scenario 1a and 1b. No collisions or jams are noticeable and the gap between the jobs is large enough to not cause any problems. In Scenario 1b, the entities are produced earlier, as specified by the *Pplan*.

Figure 5.13 compares the Oven and Freezer allocations (i.e. number of entities and temperature over time) in more detail, alongside the temperature profiles in these stations. Due to the production schedule being finished earlier in Scenario 1b, the Oven can be turned off earlier, thus preserving energy. Especially the Freezer has excessive idle time in Scenario 1a (see bottom left), which signals potential room for energetic optimization.

In Figure 5.14, the respective power supply and total energy consumptions over time are shown. The electric energy demand is noticeably lower in Scenario 1b, which can be attributed to the production line being switched off earlier in Scenario 1b after the products are finished. Heating and cooling energy are only marginally different due to

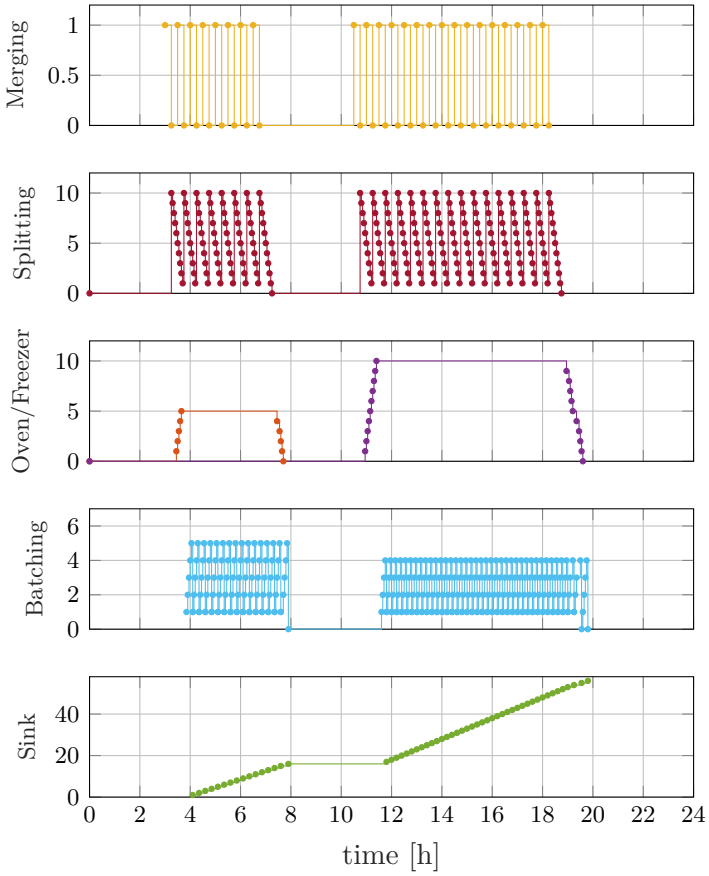


Figure 5.11: Numbers of entities over time in the different stations for Scenario 1a

the different operating times. The main energy demand stems from conditioning the thermal zones, which is the same for both scenarios [130]. Especially Zone 3 requires a constant supply of cooling power.

The temperature progression in the four thermal zones compared to the ambient temperature is depicted in Figure 5.15. While the ambient temperature changes significantly throughout the day, the zone temperatures only follow in attenuated form. The start of production at about 03:00 h produces a noticeable dent in Zone 1, after which the temperature increases more strongly due to the waste heat from the production machines. The zone temperatures are allowed to vary within a certain bandwith; the temperature control only sets in when the temperature leaves this band. Zone 3 representing the cold storage is kept at constant 4 °C. These temperatures only change marginally between Scenario 1a and 1b.

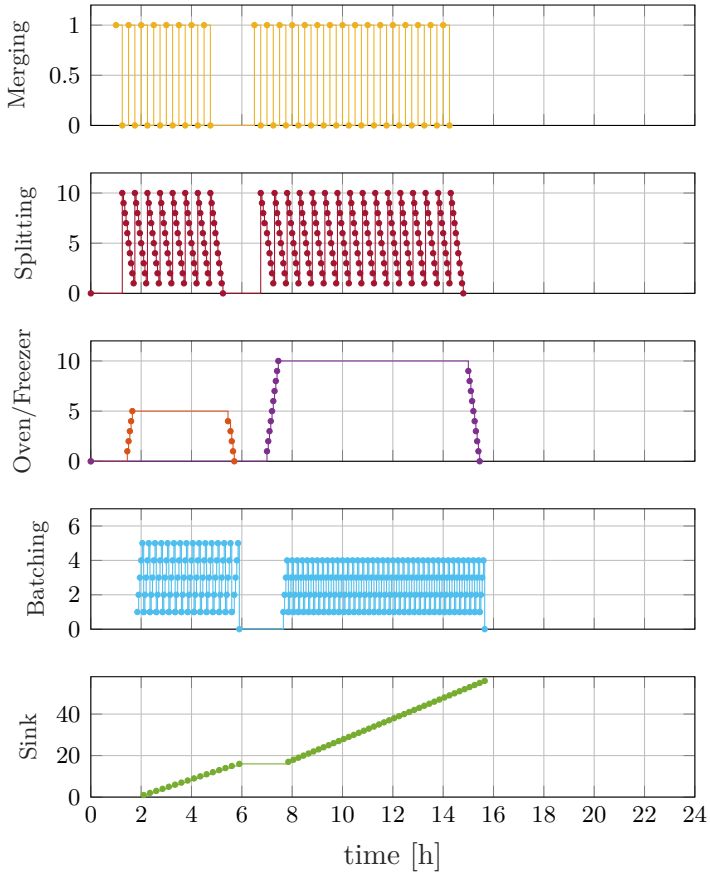


Figure 5.12: Numbers of entities over time in the different stations for Scenario 1b

5.7.3 Scenario 2

For the second scenario, we choose a *Pplan* that is more true to reality, in particular a realistic production volume is carried out over the course of one week. Figure 5.16 shows the corresponding entity flow. Overall, 15 jobs with various quantities are being executed, resulting in 381 final entities. The jobs are scheduled uniformly with enough safety gaps in between to allow changing over the whole production line to the subsequent product type. The detailed Oven and Freezer allocations are shown in Figure 5.17.

The power demand for heating, plotted in Figure 5.18 mainly follows the ambient temperature, see Figure 5.19, whereas the fluctuating electric power results from the production machines. The periodic peaks at about 15 kW stem from the energy TBS, where the Cooler produces cooling energy for the storage of the Cooling Grid (cf. Figure 5.10).

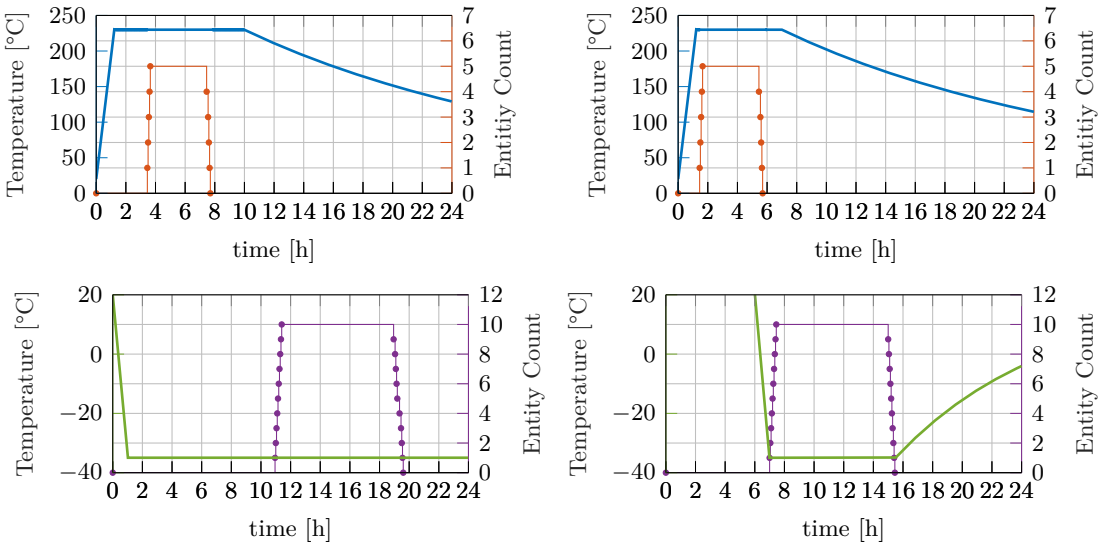


Figure 5.13: Oven (top) and Freezer (bottom) allocations for Scenario 1a (left) and Scenario 1b (right)

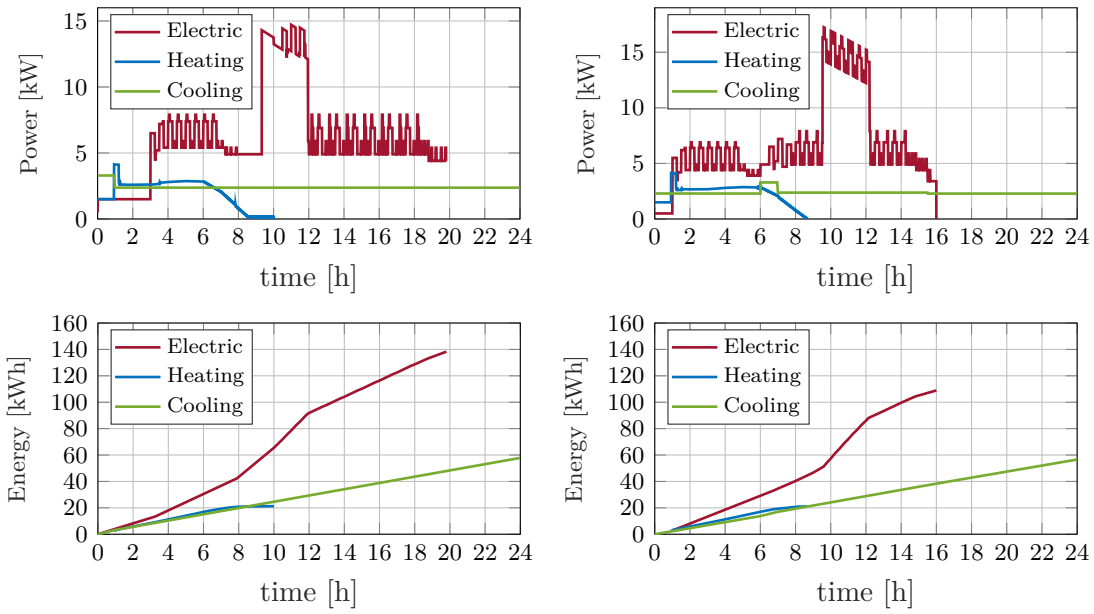


Figure 5.14: Comparison of power demand (top) and energy consumption (bottom) between Scenario 1a (left) and Scenario 1b (right)

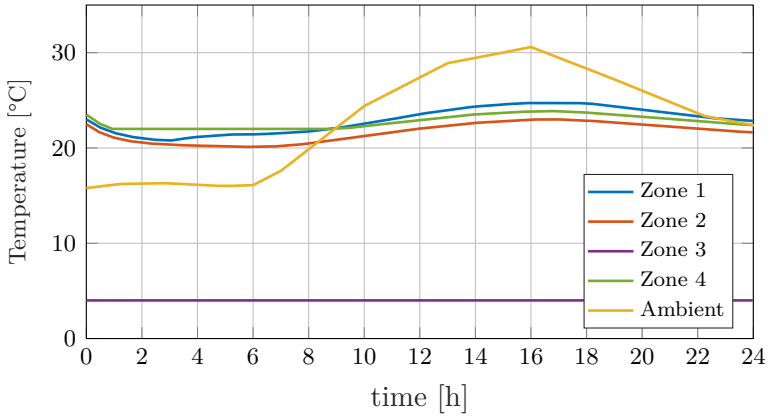


Figure 5.15: Zone temperatures and ambient temperature over time for Scenario 1a

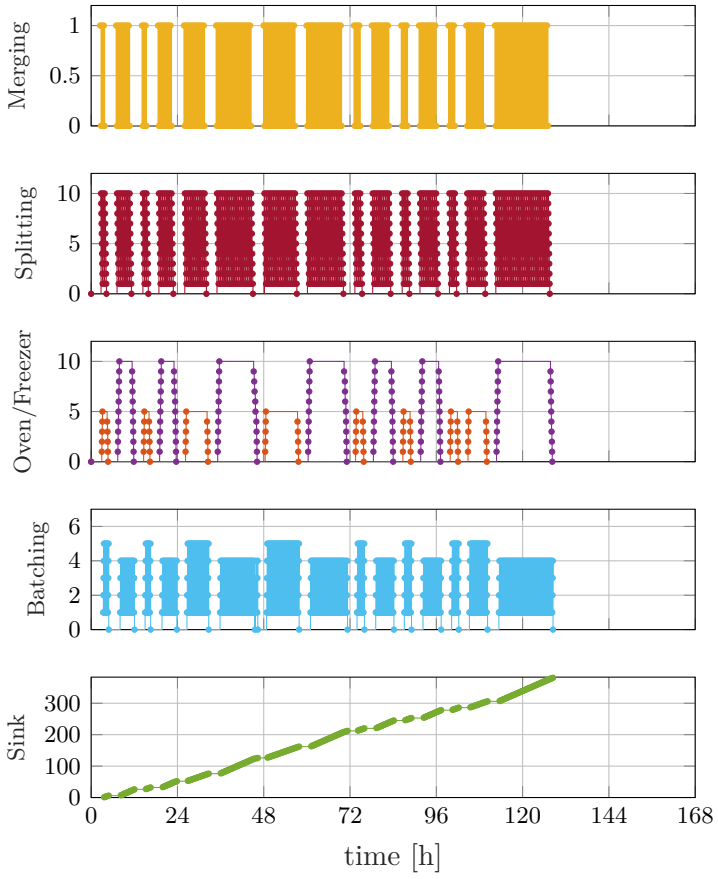


Figure 5.16: Numbers of entities over time in the different stations for Scenario 2

5.7.4 Validation

These scenarios have also been tested with the MatlabDEVS prototype implementation and are presented in [130]. They show comparable results, which also serves as a

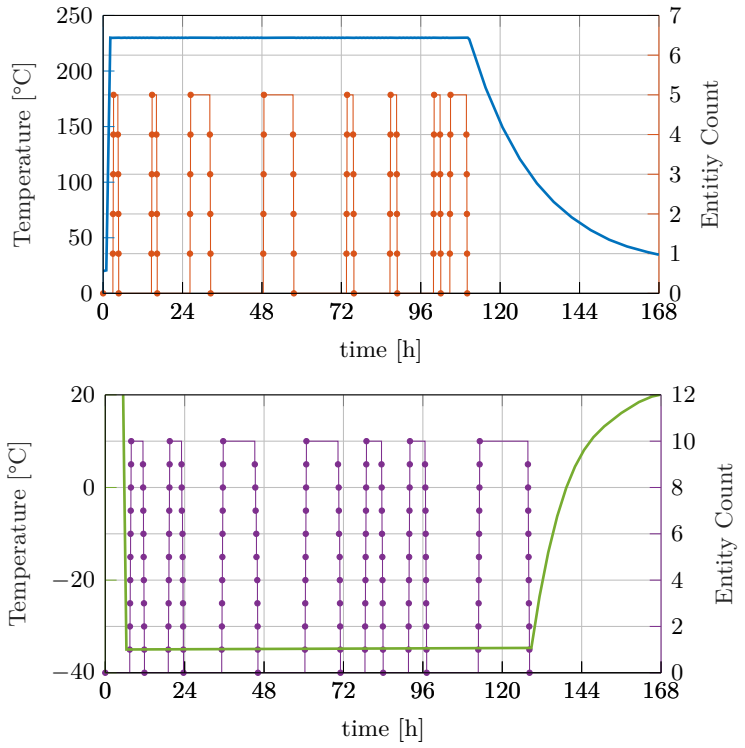


Figure 5.17: Oven (top) and Freezer (bottom) allocations for Scenario 2

verification for the stand-alone C++ implementation. Individual aspects of the overall model have been validated using independent implementations, which were developed as part of the research project BaMa [261]. The continuous sub-model including thermal zone Cubes and the energy TBS were implemented and tested using Dymola and show satisfying consistency. The Oven Cubes could also be validated against an independent implementation in Dymola, which uses native data structures for representing entities. See [224] for more details. For validating the flow of entities, entry and exit times were compared to independent calculations and also show satisfying agreement.

5.8 Case Study 2: Advanced Production Line

5.8.1 Model Description

The second case study extends the first example by modeling a realistic production line of the same industrial bakery [261, 136]. Figure 5.20 shows the conceptual model. The production includes an additional pre-proofer and three proofing cabinets with different thermal conditioning, a cooler for chilling the products after they leave the Oven, and a more advanced packaging station including cold storage and commissioning storage. The component classes are mostly the same as in the first case study, however with different

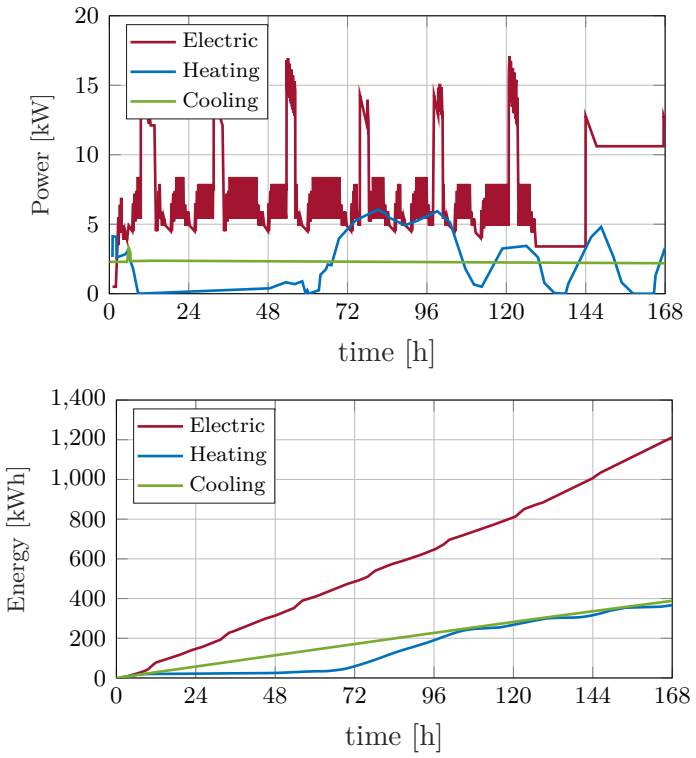


Figure 5.18: Power demand (top) and energy consumption (bottom) for Scenario 2

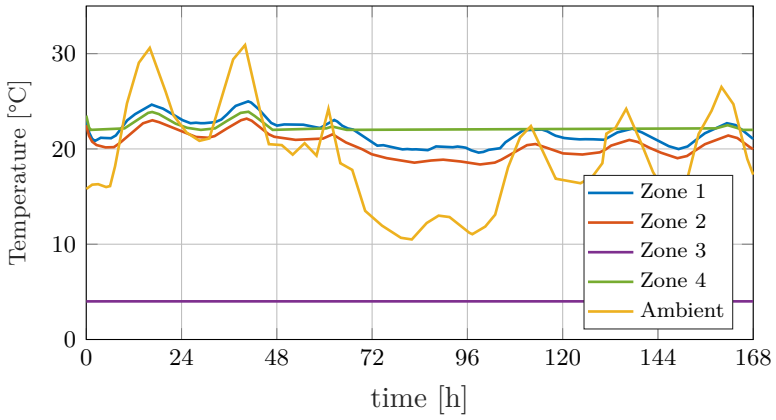


Figure 5.19: Zone temperatures and ambient temperature over time for Scenario 2

parameterizations. The proofers are of the class Production Unit. The building model with its thermal zones as well as the energy supply system remain unchanged.

The product flow is more complex compared to Figure 5.10. A total of 12 different products can be manufactured, divided into three categories that follow different paths:

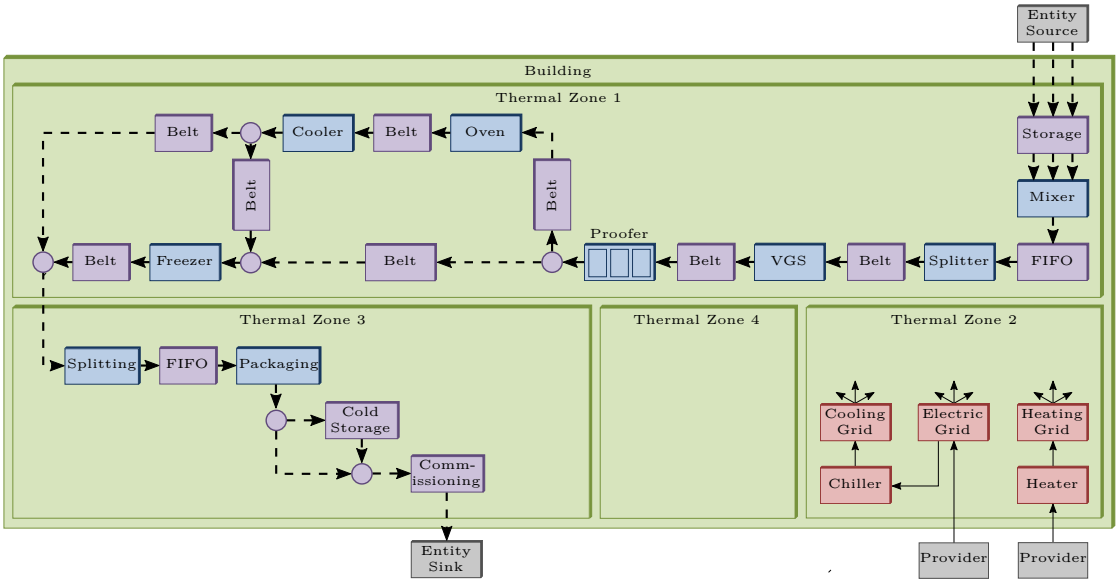


Figure 5.20: Conceptual model of Case Study 2: Advanced Production Line with production machines (blue), logistics components (purple), energy supply components (red) and thermal building zones (green). The energy supply connections for the production machines have been omitted for reasons of clarity.

1. *Fresh*: Oven → Cooler → Splitting → ...
2. *Frozen*: Freezer → Splitting → ...
3. *Partly baked*: Oven → Cooler → Freezer → Splitting → ...

The branching for the different production routes is achieved using junctions, which forward the entities differently depending on the product type (as an attribute of an entity). In addition, the third proofing zone may be omitted, thereby further complicating the material flow as the throughput times can vary greatly as a result and individual jobs can potentially "catch up" with the preceding ones.

After the pre-proofing and proofing stations, the products pass the respective Oven, Cooler and/or Freezer stations, after which the products are being separated from their Peelboards (station Splitting) before being packaged into different quantities and stored in the Cold Storage (for the frozen products) or directly in the Commissioning Storage (for the fresh products). The packaging process also takes place in a different thermal zone than the rest of the production, in order to ensure constant cooling.

All the products are being commissioned onto so-called Peelboards, which are uniform in size and contain a certain number of baking products on them (depending on the product type). A Peelboard with products is thereby represented by a single simulation entity, instead of modeling each product individually, since this level of detail is sufficient

for the material flow simulation and a higher level of detail would only have a negative effect on the simulation performance. In this context, investigations were carried out by means of comparative simulations (in Plant Simulation), which show a performance gain through this simplification of factor 60.

A further model simplification and resulting packaging limitation is that the packaging ratios must always be integer so that no excess entities are left over. The model does not take into account the rejection of unfinished or remaining entities. The raw material storage at the beginning of the production line takes into account the five quantitatively most important ingredients (flour, water, etc.), which are supplied by external providers (i.e. Entity Sources).

5.8.2 Validation

The aim of this case study is to provide real-life tangible results that are applicable to the real production line, which is housed at one of the project partner's production sites. To this end, an extensive validation of the model and its parameters was carried out. Measurement data were collected on the real plant during production over the course of 10 days, especially on the power consumption of the individual stations during production, temperature data and cooling curves. These were then aligned with the actual operating data (e.g. number of units produced, real start and end times) to obtain a consistent overall picture of production. A portion of the measurement data was then used to calibrate the model parameters, while another part was used as test data for validation. More details can be found in [29, 261, 263]. The validated model therefore provides realistic comparisons of different production scenarios (Production plans, process parameters, etc.) and allows to assess the associated energy consumption.

Case study scenarios have been carried out with production plans spanning one, 7 and up to 30 days. In the longer scenarios, a simulation run comprises up to 500,000 entities, while simulation runtime of a typical 7-day scenario is about 105 sec. The simulation runs could be scaled up without problems and demonstrate practical applicability on a real-world scale. Further details are given in [261].

5.9 Discourse

In the following, we want to discuss some of the issues and findings we have encountered during modeling and hybrid simulation with hyPDEVS.

Coupled Behavior: According to Equation (5.3), Coupled hyPDEVS cannot have states of their own (like S for Atomics); all model states must be encapsulated inside an Atomic component. This may sound trivial, but it has an impact on the way hierarchical modeling can be achieved. For example, in a real production facility, the production machines are housed within a room, which can be described as a thermal zone. If one wants to retain this topology in the simulation model, as we have done in the case

studies using the Thermal Zone Cubes, the thermal zone has to be realized as a Coupled hyPDEVS. However, since the thermal zone also has a (continuous) dynamic behavior of its own (in particular the heat exchange through the walls and with its internal loads), one also has to include an additional Atomic hyPDEVS that implements this behavior.

Closure under Coupling: The closure under coupling property of hyPDEVS is the reason it is possible to develop modular component-based models in the first place, which are also hybrid (at the Atomic level). This is an important feature of DEVS based modeling and provides a foundation for research into related aspects, such as consistency or formal verification [319].

Maintaining the Model: One important advantage of the hyPDEVS approach and its tighter integration of discrete and continuous model parts, especially compard to co-simulation becomes apparent when trying to modify the model. For example, if one wanted to exchange the (continuous) PI controller in the Oven Cube with a (discrete) two-point controller, this would require significant effort and relocating interfaces in the case of co-simulation because the controller suddenly has to move from the continuous sub-model to the discrete one. In the hyPDEVS model on the other hand, exchanging the controller is no problem and does not change any interfaces.

CHAPTER 6

Multi-objective Production Planning Optimization

In the previous chapters, we have described a method for simulating production systems and their energy demand as well as the simulation implementation and demonstration by means of case studies. In this chapter, we focus on developing a method for optimizing production scheduling that employs these simulations for evaluating different production scenarios regarding their energy demand as part of a multi-objective target system. Together, this forms a simulation-based optimization approach for energy-aware production scheduling to aid modern industrial PPC and APS tools.

6.1 Introduction

The goal of PPC optimization is to find an optimal combination of production resources, such as equipment, utilities or material, to achieve a given production target in the best possible way. Operative Production Planning and Control (PPC) methods consider production resources and cost factors to find optimal production schedules [161]. The challenge is to find the best configurations among all feasible solutions that lead to the optimal performance without explicitly evaluating each possibility [53]. The outcomes are production plan configurations specifying at what time which jobs in which quantities are best released for production [129].

We focus here on production planning (with sequencing and scheduling) in flow shop production layouts with the aim to increase energy efficiency and reduce energy costs. There are of course a lot of other ways to improve energy efficiency of production systems, both in design and operation. Aside from the production itself, especially the operation of the Technical Building Services (TBS) offers a lot of additional potential for energy savings [167], not at least because of the building HVAC accounting for a large portion

of a company's total energy demand [261]. For example, it would be possible to optimize the filling of thermal energy storages at times when energy costs are low (e.g. at night, or when renewable energy sources are available), or to improve the operating behavior of chillers suppling cooling energy [157].

However, the operation optimization of TBS is a separate topic and should be excluded from our consideration in order to keep the problem manageable. While TBS and their energy turnover are included in the simulation and therefore contribute to the overall energy demand, no specific TBS-related decision variables are considered.

That being said, we do include time-dependent energy pricing into the target system. Especially large companies with high electricity consumption, e.g. in the steal production industry, usually handle their energy purchases on the electricity exchange spot market [26] (using day-ahead as well as intra-day trading) where electricity prices may fluctuate strongly throughout the day, depending on supply and demand, especially in the presence of volatile renewable energy sources [160]. This can have a significant impact on the planning result, as we will demonstrate in the case study.

There have been other related works conducted in the context of the same research project that employ the same hybrid simulation that has been developed here. A similar optimization strategy was developed based on a Genetic Algorithm, where the author of [261] describes the general concept.

The remainder of this chapter is structured as follows: Section 6.2 elaborates in more detail on the simulation-based optimization strategy. Section 6.3 formulates the multi-objective optimization problem, system under study and decision variables while Section 6.4 describes the proposed optimization algorithm. In Section 6.5, we demonstrate the feasibility of this method on a flow shop scheduling problem of an industrial bakery, where we compare different scenarios and highlight the potential benefit of considering energy as an optimization target. The chapter concludes with a discourse in Section 6.6.

6.2 Simulation-based Optimization Strategy

6.2.1 Meta-heuristics

The dynamic simulation serves as a prescriptive tool to predict and evaluate the performance of a given scenario. Dynamic simulation allows to consider more complex systems than with conventional analytical models, especially for highly time-dependent problems, while offering more accurate predictions and overall improving planning quality.

Instead of a closed mathematical formulation of the target function and all necessary constraints (like they are used with other approaches such as MIP), the constraints are realized partly implicitly by means of the simulation. Only by computing the simulation can it be evaluated whether or not the solution is truly feasible. Nonetheless, additional constraints may still be formulated within the target system, e.g. as soft constraints, to better guide the search and thus improve optimization performance.

However, the fact that the overall objective function – or its derivatives – is usually impossible to define in a closed analytical form prevents straightforward deployment of many standard optimization algorithms. Instead, many practical simulation-based optimization solutions employ meta-heuristic methods that rely solely on the evaluation of the objective function itself and do not require the exact closed form objective function to traverse the search space [149]. These algorithms modify a candidate solution in an iterative manner to find a near-optimal solution until termination criteria are met.

In fact, one would arrive at the same conclusion that meta-heuristics are a viable option when ignoring the simulation and just looking at the problem complexity itself: In general, flow shop scheduling problems, especially for larger real-world instances, are well-known in operations research to be NP-hard problems [309, 154], implying that they cannot be solved in polynomial time. Therefore, exact optimization methods are not feasible in general and approximative heuristic approaches are preferred for practical applications.

From an optimization standpoint, the complexity of the underlying optimization problem with multiple competing objectives and complex constraint conditions also drives the need to use simulation methods for evaluating the target system. At the same time, these methodologies are being enabled by recent advances in computational techniques that allow to perform increasingly complex computations in feasible time [272].

Meta-heuristic methods are common in application-oriented operations research as they are able to deliver acceptable solutions with feasible time and resources even for larger problem sizes, which is advantageous for the scalability of our method. Even if the solution found may not be the global optimum, it is often good enough for practical applications.

According to [295], meta-heuristics and customized multi-objective heuristic approaches are well-suited for applications in real-life industrial production planning problems, in contrast to exact approaches that usually require simplified models. Meta-heuristics allow to explore the search space more efficiently and effectively, especially if they are tailored to the individual problem [119]. Among these, Variable Neighborhood Search (VNS) algorithms have shown excellent capability for solving scheduling problems [238]. This is in accordance with other publications, e.g. [312, 238], which have successfully applied VNS for job scheduling problems in the production domain. In [106], the authors compare different optimization methods for simulation-based optimization of production plans, in which VNS also leads to the best results.

6.2.2 Simulation-Optimization Cycle

The iterative evaluation of possible solution candidates performed by the meta-heuristic creates an cycle of alternating simulation and optimization computations as the core of the simulation-based optimization methodology. The overall iteration cycle as an overview is illustrated in Figure 6.1. The starting point is a given demand plan (*Dplan*) specifying how many of which entities (i.e. products) need to be delivered when. This *Dplan* serves as a basis to generate an initial solution.

The optimization algorithm employs its own encoding of the optimization problem, usually as a vector x of decision variables, which have to be translated[1] into a representation that is understandable for the simulation, i.e. a *Pplan*. This may be complemented by other *Pplans* and other control signals, which do not represent independent decision variables and thus can be derived from the existing variables in x, but are necessary input for the simulation. After running the simulation, the simulation results (calculated energy demand, job completion times, etc.) are fed back to the optimization to be evaluated for its fitness using a specified cost function $f(x)$ (objective function). Based on this evaluation, the optimization algorithm iteratively adapts the solution and repeats the cycle in order to improve the solution, until certain termination criteria (e.g. fixed number of iterations, computation time threshold) are met.

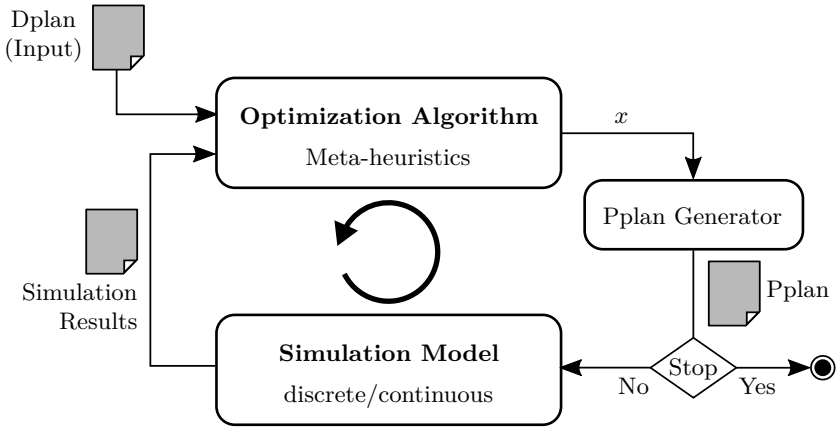


Figure 6.1: Simulation-optimization cycle. Starting from a demand plan (*Dplan*) of entities to be produced, the optimization algorithm generates a production schedule (*Pplan*) from the decision variables x and adapts it iteratively based on an evaluation of simulation results.

Meta-heuristic methods in the context of simulations-based optimization are also referred to as simheuristics [151, 63, 116]. Simheuristics have applications for combinatorial optimization problems in different fields, including scheduling, manufacturing or vehicle routing.

6.3 Problem Description

Based on the overview of the simulation-based strategy presented in the last section, we now dive into the optimization method itself. First, we formalize the production system and the optimization problem in more detail.

[1] In the context of Genetic Algorithms (GAs), these encoded and decoded variables are typically called genotype and phenotype, respectively [65].

The optimization problem deals with sequencing and time scheduling a given list (*Dplan*) of n_{jobs} jobs to be produced by a production system while minimizing energy demand together with other production goals as part of a multi-objective problem. This includes optimizing setup times to reduce energy demand and increase production throughput. The optimization goals may differ from one application to the other in order to be adjusted to different production systems and optimization targets. The production system itself, in this case, is given as a simulation model, like described in Chapter 5.

6.3.1 Multi-objective Problem Formulation

In general, multi-objective optimization as part of the area of multi-criteria decision-making is the process of selecting the best possible course of action from all the available alternatives while attaining to more than one objective or goal[2] and satisfying a set of constraints dictated by the environment, process and resources [143]. Multi-objective optimization problems can be formalized[3] as

$$\min_{x \in \mathcal{X}} f(x) = \min_{x \in \mathcal{X}} \left(f_1(x), f_2(x), \ldots, f_k(x)\right) \tag{6.1}$$

where x is the (n-dimensional) vector of decision variables and f describes the vector-valued objective function, also called target system,

$$f : \mathcal{X} \to \mathbb{R}^k, \quad f(x) = \left(f_1(x), f_2(x), \ldots, f_k(x)\right), \tag{6.2}$$

which is comprised of k objectives or sub-goals f_i. The set $\mathcal{X}$ is the set of feasible solutions, which is defined by m constraint functions

$$g_j(x) \leq 0, \qquad \forall j = 1, \ldots, m, \tag{6.3}$$

any of which may be non-linear [143]. In addition to these hard constraints, which are strictly required by the decision variables to be satisfied in order to constitute a feasible solution, so-called soft constraints can be added in the objective function that penalize conditions which are undesirable.

Using soft constraints can have advantages in practice as sometimes constraints are not always so strict. Some of them might have some leeway, e.g. delivery penalties might be acceptable as long as the benefit gained on the other side still pays off for the company. This must be considered on an individual basis and the soft constraints with penalties provide a lot of flexibility to adjust the constraint system to the individual needs.

The sub-goals typically constitute conflicting targets, implying balancing necessary trade-offs between them. These sub-goals may represent energy costs, or storage costs, delivery tardiness, production utilization or lead time. For assessing energy costs, the temporal

[2] Technically, a slight distinction could be made between the terms *objective* and *goal*. While objectives give the desired "direction" in which to look, goals give a desired target level to achieve [143]. However, we will not insist on this distinction here and instead use the terms interchangeably.

[3] In this case as a minimization problem.

variability of energy prices might need to be considered in order to get an accurate picture. This is also investigated in Section 6.5.3. In addition to the basic energy price assessment, other energy-related aspects can be included as well, for example penalties for load peaks [261]. This would favor production schedules that smooth out load peaks and thus contribute to a more even energy utilization.

Most often, the minimization in multi-objective optimization is understood in the sense of Pareto optimality (also called Pareto efficiency) [90]. A solution $x \in \mathcal{X}$ is called Pareto-optimal (or efficient) if there does not exist another feasible solution $x' \in \mathcal{X}$ such that $f_i(x') \leq f_i(x)\ \forall i = 1, ..., k$ and strict inequality for at least one f_i. In other words, none of the objective functions can be improved without degrading some of the other objective values. The corresponding vector $f(x)$ is then called non-dominated and the set of all Pareto optimal solutions is often referred to as Pareto frontier.

Without additional information, all Pareto optimal solutions are considered equally good. As a decision-support tool in practice, a single solution needs to be found, which can be achieved by adding subjective preference information of the decision maker. One common approach for this is discussed in Section 6.4.2.

The fact that the underlying simulation model involves continuous as well as discrete state variables might suggest to also employ hybrid or mixed-integer optimization methods. However, not only would this substantially increase the problem complexity, the thus obtained increase in solution accuracy is unnecessary for most production scheduling problems. We instead discretize the search space by only using discrete shifting intervals to simplify the problem. A typical granularity of 15-minute intervals is more than sufficient for our application. In fact, making the coarseness adjustable gives flexibility to the user to balance search effort against solution quality.

6.3.2 System under Study

In Chapter 5, we have already seen some examples of production systems that are our subject of study. Just to reiterate and give a more comprehensive overview, Figure 6.2 depicts a representative production system in flow shop layout. It consists of a several production stations $s \in \mathit{Stations}$ (including logistics components), entity sources $r \in \mathit{Sources}$ and entity sinks $k \in \mathit{Sinks}$, as well as energy supply systems with thermal energy storage $e \in \mathit{EnStorage}$ and several thermal zones $z \in \mathit{ThermZones}$.

The entity sinks provide as simulation result numbers of entities $n_{k,t,l}$ of type l, which arrived at time t in the sink k. But not only the sinks, but each of the production stations, and especially the storage components, may hold entities, which are still present at the end of the observation period, i.e. when the simulation terminates. While these stored entities can potentially also be included in the evaluation (e.g. as partially finished products), we consider entities to be finished only after they have arrived in one of the entity sinks. The production jobs themselves are not associated with a specific customer order, meaning that it is not relevant which customer gets which specific product, only the *type* of the product being produced is relevant.

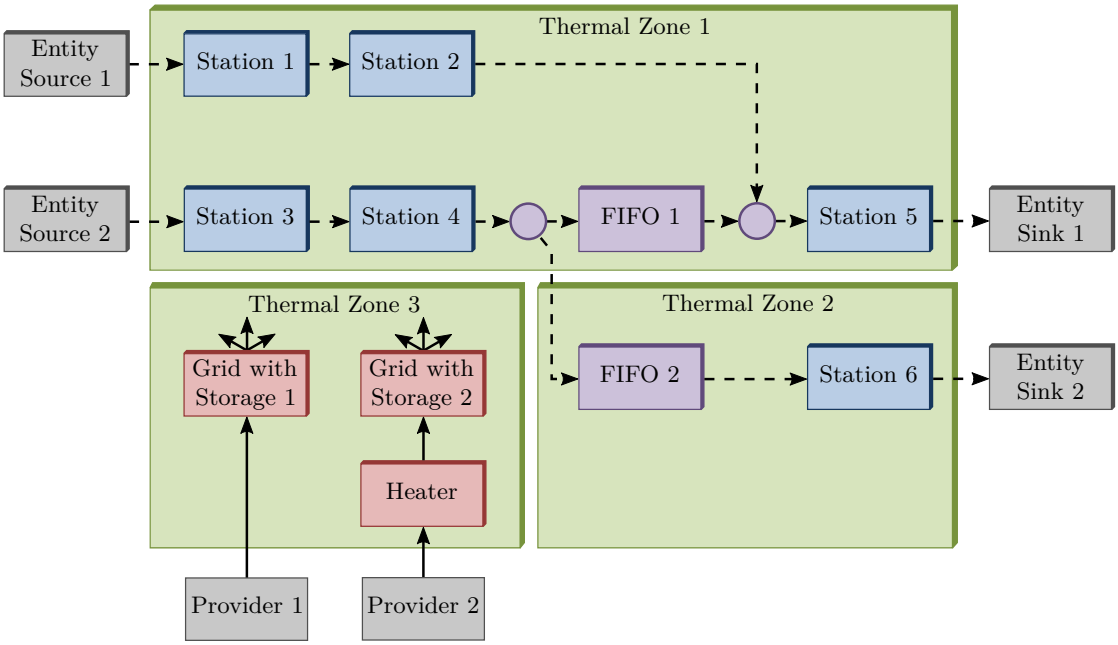


Figure 6.2: Exemplary production system layout with production stations, entity sources and sinks, energy supply systems and thermal zones

Moreover, a set *Prov* of energy providers supplies energy from the outside through the system boundary. During the simulation, each energy provider s computes the power supply $P_{s,t}$ over time t, which is then used by the optimization for evaluating the energy costs in the target function. This is because the energy that crosses the system boundary is also the one that is being billed to the energy consumer.

6.3.3 Decision Variables

The decision variables x can be written as follows:

$$x = (x_{\text{start}}, x_{\text{setup}}), \tag{6.4}$$

$$x_{\text{start}} = (x_1, x_2, \ldots, x_{n_{\text{jobs}}}), \tag{6.5}$$

$$x_{\text{setup}} = (x_{\text{setup},n,s})_{n \in \text{Dplan},\, s \in \text{SetupStations}} \tag{6.6}$$

These optimization variables define the production schedule (*Pplan*), which describes which production job should is to be produced when and in which quantity. Since we focus on flow shop scheduling, only the starting times $(x_1, x_2, \ldots, x_{n_{\text{jobs}}})$ for each jobs are independent decision variables, not the times on each individual station. The *Pplans* of other stations are then oriented towards the starting times. In addition to the starting time for each job, x also includes setup time durations[4] $x_{\text{setup},n,s}$ of relevant production

[4]Instead of the setup time duration (as a relative value), the (absolute) starting times at these stations could also be used as decision variables. However, the durations are a bit easier to handle in the

stations $s \in SetupStations \subseteq Stations$ with $|SetupStations| = n_{\text{setup}}$. Together, these decision variables allow to optimize job sequencing and scheduling as well as setup processes, all of which have an influence on the energy demand.

In the same way, the starting times at each individual station could also be included as decision variable in x. This would be the more generic variant that would allow to also model job shop scheduling applications. For flow shops, however, these individual starting times cannot be chosen independently if a constant material flow is to be maintained, and this would in turn require strong constraints to model these dependencies. The increased solution space (of the more generic approach) would thus be strongly reduced again and the efficiency of the search would depend on the optimization procedure and how well it can handle the constraints.

Moreover, shut-off times of the stations could also be controlled from the optimization. The simulation itself allows to specify shutdown events in the *Pplan*. This is especially useful for thermal processes, like an oven, in order to save energy. However, experiments have shown that including shutdown times as decision variable in x has no significant advantage and would only unnecessarily inflate the search space and thus the computing time. Energetically, it usually makes more sense to either shut off immediately after the end of a job, or not at all in order to wait for the next job. On top of that, the shutdown time can usually only be varied within a very limited time interval, namely between the end of the last job and the beginning of the next job. If two jobs follow directly after one another, the shutdown effect disappears completely.

Based on these considerations, we have taken a heuristic approach for choosing the shutdown times. If a job is not followed by another job for a certain (parameterizable) time t_{idle}, a shutdown event is added to the *Pplan* immediately following this job.

6.4 GVNS Optimization Method

We employ a single-solution meta-heuristic method based on a General Variable Neighborhood Search (GVNS) procedure. The VNS [238] is an effective method guiding a local search by switching between increasingly larger neighborhoods to efficiently explore the solution space. The VNS consists of

1. a shaking phase that generates randomized perturbations of the solution for diversifying the search to escape local optima, and
2. an intensification phase that searches for improvements in the local neighborhood, typically using a local search procedure.

For improving the performance of the search, the Generalized VNS (GVNS) replaces the local search by a Variable Neighborhood Descent (VND) procedure [251, 126]. The

implementation, since they remain unchanged during shifting and switching of jobs.

Variable Neighborhood Descent (VND) is similar to the VNS, but does not include shaking and thus limits the search to strict improvements (i.e. descent). It does not use the same neighborhood structures as the VNS, which offers more flexibility in tailoring the search procedure to the problem instance.

Algorithm 6.1 presents an overview of the optimization method as pseudocode. After generating a feasible initial solution x, the VNS component (described in more detail in Section 6.4.3) guides a search for improvements over a fixed number of iterations n_{iter} by generating in each iteration a random neighboring solution x' (by means of the neighborhood structure $\mathcal{N}_k(x)$) that serves as starting point for the VND that tries to find a local minimum x''. If the solution is accepted, the VNS restarts with neighborhood $\mathcal{N}_1$ and the new starting solution x''. Otherwise, the neighborhood is enlarged. The acceptance criterion is based on Simulated Annealing (SA) [251] for further diversifying the search by allowing to accept potentially worse solutions during the early search phase. Infeasible solutions are also allowed during the search and are evaluated based on a penalizing cost function, which is described in Section 6.4.2. More details are described in the following sections.

Algorithm 6.1: GVNS/SA algorithm

```
1  N_k ← set of VNS neighborhood structures for k = 1, ..., k_max;
2  x ← InitialSolution();
3  k ← 1;
4  i ← 0;
5  T ← T_0;                                          // SA temperature
6  while i ≤ n_iter do
7  |   x' ← shaking(N_k(x));                        // generate random perturbation
8  |   x'' ← VND(x', k);                            // local improvement search
9  |   if acceptSA(x'', x, T) then                  // acceptance based on SA
10 |   |   x ← x'';
11 |   |   k ← 1;
12 |   else
13 |   |   k ← (k mod k_max) + 1;                   // try next neighborhood
14 |   end
15 |   i ← i + 1;
16 |   update T;
17 end
```

6.4.1 Initial Solution

An initial solution (function $InitialSolution()$ in Algorithm 6.1) is generated by employing a custom construction heuristic, which takes the jobs from the provided input list ($Dplan$), sorts them in increasing order of due date and then schedules the jobs in this order as

soon as possible (forward scheduling) while considering safety gaps between jobs as well as fixed setup times. This method ensures that no collisions occur, however, further constraint violations, such as delivery delay, or energy considerations, are not checked during this phase.

Note that setting up a *Dplan* might include an initial heuristic (done manually by the user) on how to define job quantities (i.e. batch sizes) meaningfully. Since these are used to generate the initial solution, they also influence the efficiency of the search.

6.4.2 Generalized Cost Function

The algorithm allows infeasible solution during the search process. Constraints are penalized by means of the generalized cost function

$$f(x) = \omega_1 \cdot f_{\text{dev}}(x) + \omega_2 \cdot f_{\text{en}}(x) + \omega_3 \cdot f_{\text{st}}(x) + \omega_4 \cdot f_{\text{del}}(x) + \omega_5 \cdot f_{\text{sep}}(x) \tag{6.7}$$

where f_{dev} denotes a penalization for deviating number of entities produced, f_{en} the energy costs, f_{st} the storage costs, f_{del} is a penalization for any potential delivery delay, and f_{sep} penalizes job separation violations. These part goals are evaluated based on the simulation results. The coefficients $\omega_i > 0$ are weighting factors, which may be adapted by the decision maker to balance their preferences for individual part goals and trade-offs in a transparent manner. This *weighted sum* method widely used in multi-objective optimization problems in practice [143] as it is easy to implement and intuitive for the user. However, the results are often highly dependent on the weights [102].

The individual part goals are calculated as follows:

Entity Deviation: The entity deviation f_{dev} simply compares the actual number of entities produced n_{act} to the target entities n_{target} for each product type l given in the input list (*Dplan*) and penalizes the deviation with a factor ρ_{dev}:

$$f_{\text{dev}}(x) = \sum_{l} |n_{\text{act},l}(x) - n_{\text{target},l}(x)| \cdot \rho_{\text{dev}}, \tag{6.8}$$

where n_{act} results from the simulation by summarizing all sinks,

$$n_{\text{act},l}(x) = \sum_{k \in \text{Sinks}} \sum_{t_i} n_{k,t_i,l}(x), \tag{6.9}$$

and n_{target} is given from the *Dplan* by summarizing all order lots:

$$n_{\text{target},l} = \sum_{t_j} \text{Dplan}_{t_j,l}. \tag{6.10}$$

The same result can be obtained by looking at the stock level from Equation (6.13) (see below), which has to amount to zero in the end, i.e. $S_{t_{\text{end}},l}(x) = 0$. Choosing a large value for ρ_{dev} strongly avoids any deviation in the number of entities as this would not constitute a desired solution.

Energy Costs: The energy costs f_{en} are calculated based on the simulation results, by taking the simulated power supply P_s from each of the external providers $s \in \text{Prov}$, rate them with time-dependent energy prices c_s and accumulate them over time to obtain the overall energy costs:

$$f_{\text{en}}(x) = \sum_{s \in \text{Prov}} \sum_{t_i} P_{s,t_i}(x) \cdot (t_{i+1} - t_i) \cdot c_s(t_i). \tag{6.11}$$

This method of using variable energy prices allows to take into account different effects that have an influence on the energy price and therefore may potentially influence the production planning result, like for example lower energy prices during nighttime, or an additional photovoltaic system that provides solar energy during the day.

Storage Costs: The storage costs f_{st} can be calculated in different ways, either on a per-job basis where the job completion times are used, or on a per-type basis, where only the number of products produced of each type is relevant. The first variant suitable for job shop layouts where each job is assigned to a specific customer, the second variant is easier for flow shop production without specific customer allocation and variable lot sizes (e.g. by merging jobs). The latter results in

$$f_{\text{st}}(x) = \sum_{l} \sum_{t_i} \max\{S_{t_i,l}(x), 0\} \cdot (t_{i+1} - t_i) \cdot c_{\text{st}}(t_i) \tag{6.12}$$

with the (potentially time-dependent) storage price per time unit and quantity c_{st} and $S_{t_i,l}$ the current stock level at time t_i for type l, which can be calculated as

$$S_{t_i,l}(x) = \sum_{k \in \text{Sinks}} \sum_{t \le t_i} n_{k,t,l}(x) - \sum_{t \le t_i} \text{Dplan}_{t,l} \tag{6.13}$$

This equation calculated the stock level based on the simulation results, particularly the number of entities of type l arriving at the final sinks $k \in \text{Sinks}$ at time t. Alternatively, the progression of the stock level can be obtained directly from the simulation when using storage components in the simulation.

Delivery Delay: The stock level $S_{t_i,l}$ is also useful for finding potential delivery delays, namely by looking for times when the stock level is negative:

$$f_{\text{del}}(x) = \sum_{l} \sum_{t_i} -\min\{S_{t_i,l}(x), 0\} \cdot (t_{i+1} - t_i) \cdot \rho_{\text{del}} \tag{6.14}$$

with ρ_{del} as the delay penalty. Alternatively, calculating the delivery delay costs for job shop production can be done similarly to calculating the storage costs by looking at the job completion times and comparing them to the *Dplan* due dates.

Separation Constraint: The last element $f_{\text{sep}}(x)$ the cost function Equation (6.7) specifies a penalty for violating the constraint that, in each station, between one job an the next, the station must be cleared of entities:

$$f_{\text{sep}}(x) = \sum_{s \in \text{Stations}} \left| n_{\text{jobs},s}(x) - \sum_{j} useStation_s(\text{Dplan}_{t_j,l}) \right| \cdot \rho_{\text{sep}}. \tag{6.15}$$

Hereby, $n_{\text{jobs},s}$ is the number of jobs that have been processed at station s and results from the simulation, while ρ_{sep} again denotes a penalty factor. The function $useStation_s$ returns whether or not the respective job from the *Dplan* uses Station s. This soft constraint was introduced partly due to restrictions in the simulation itself, but does also have merit in the real world. Without this constraint, individual jobs would be pushed together to such an extent that they overlap in individual stations, thereby preventing correct setup taking place for the following job.

6.4.3 Neighborhood Structures

The general operation of the VNS has been described in Algorithm 6.1. It is responsible for diversifying the search in a structured way during the shaking phase by successively changing the neighborhood structures $\mathcal{N}_k$. These neighborhood structures are usually defined implicitly by means of operators that modify the solution. For our case, we have defined four different operators:

1. *OpSwitch:* This operator changes the order of jobs by taking a random number r of successive jobs (starting from a random position) and moving them to a different position. Hereby, r is chosen in the interval $r \in [1, \min\{r_{\max}, n\}]$, where n is the overall number of jobs and $r_{\max}$ changes depending on the neighborhood k, see Table 6.1.
2. *OpShift:* The shifting operator takes a random position and moves all subsequent jobs by a specified time t_{shift}, where t_{shift} depends on k.
3. *OpChangeSetuptime:* Here, the setup time is changed by a specified amount t_{setup}.
4. *OpMerge:* The merging operator takes two random jobs, which are removed by a distance of d, and which have the same product type and combines them into one job. Merging jobs has the advantage that it reduces the number of setup processes and avoids gaps between jobs, thereby increasing production and energy efficiency.

The parametrization of the neighborhood structures, as presented in Table 6.1, has been calibrated through various scenarios. They can be adjusted, however, and tuned to specific problem instances. This in fact allows a lot of flexibility for the user to tailor the search to the specific problem and influence the performance of the search.

Despite the large number of parameters used in the GVNS heuristic, it is relatively easy to find a set of parameters that works well for different use cases. The parameters have

Table 6.1: Neighborhood structures used in the GVNS

k	Operator	Shaking	VND
1	1	$r_{\max} = 2$	$r_{\max} = 1$
2	1	$r_{\max} = 4$	$r_{\max} = 1$
3	2	$t_{\text{shift}} = 8\,\text{h}$	$t_{\text{shift}} \in \{4, 2, -1, 0.5\}\,\text{h}$
4	2	$t_{\text{shift}} = 12\,\text{h}$	$t_{\text{shift}} \in \{4, 2, -1, 0.5\}\,\text{h}$
5	3	$t_{\text{setup}} = 0.5\,\text{h}$	$t_{\text{setup}} \in \{0.5, 0.25\}\,\text{h}$
6	4	$d = 2$	$d = 1$

been tuned in a systematic fashion following the approach described in [237, 246] using multiple different scenarios. Starting from a base parameter setting found during the development of the GVNS method using trial end error, the parameters were varied consecutively, always keeping the best setting and proceeding with the next one.

6.4.4 Variable Neighborhood Descent

As mentioned, the VND is used in place of a local search procedure inside the VNS to improve the generated solution in the intensification phase [126]. The basic operation is sketched out in Algorithm 6.2.

Algorithm 6.2: VND algorithm

```
1  N_{k,l} ← set of VND neighborhood structures for l = 1, ..., l_max;
2  l ← 1;
3  while l ≤ l_max do
4  |   find x'' ∈ N_{k,l}(x) with f(x'') ≤ f(x')  ∀x' ∈ N_{k,l}(x);
5  |   if f(x'') < f(x) then
6  |   |   x ← x'';
7  |   |   l ← 1;
8  |   else
9  |   |   l ← l + 1;                    // try next neighborhood
10 |   end
11 end
```

The neighborhood structures are based on the same operators as described in Section 6.4.3, albeit with different parameters, see Table 6.1, and depend on the current VNS neighborhood $\mathcal{N}_k$. For the switching operator, the VND neighborhood containing all pairs of successive jobs is explored exhaustively. The same is done with respect to merging. For *OpShift*, the VND checks shifting groups of jobs in a binary search pattern by a set of different times t_{shift}. Similarly for the setup times, which are being reduced iteratively by a set of different values for t_{setup}.

6.4.5 Acceptance Criterion

Instead of accepting only improving solutions, a modified acceptance criterion ($acceptSA()$) is used that is based on a Simulated Annealing (SA) method [251]. It enables to further diversify the search and better escape local optima by accepting potentially worse solutions, albeit with decreasing probability as the search progresses [108]. To be more precise, while an improving solution is always accepted, a deteriorating solution x'' is accepted with the probability

$$p_{\mathrm{SA}} = e^{-(f(x'')-f(x))/T}, \tag{6.16}$$

where f is the generalizes cost function (see Section 6.4.2) and T is the Simulated Annealing (SA) temperature that decreases linearly after every VNS iteration in such a way that $T < 10^{-3}$ during the last 10% of iterations. This effectively tightens the acceptance criterion as the search progresses until, finally, only improving solutions are accepted at the end. The temperature T is initialized with T_0 according to

$$T_0 = -\frac{\Delta_{\mathrm{SA}}}{\log(0.5)}, \tag{6.17}$$

meaning that, initially, a solution being Δ_{SA} (deterioration percentage) worse than $f(x)$ is accepted with a probability of 50%. Experiments revealed a value of $\Delta_{\mathrm{SA}} = 20$ to be suitable for various case studies.

6.5 Case Study Experiments

This section demonstrates the application of the GVNS/SA optimization method on a flow shop scheduling case study as a proof of concept, for which the Case Study 1 (Simple Production Line) from Section 5.7 was chosen. This simple case study keeps the complexity manageable in order to be able to manually verify the optimization results and check for correct behavior of the algorithm. The optimization was implemented as a prototype in MATLAB and coupled with the standalone hyPDEVS simulation.

6.5.1 Implementation

Just to recap the simulation implementation: The production stations, especially Oven and Freezer, are modeled as hybrid discrete/continuous models, where the discrete model is responsible for entity flow and control logic, and the continuous model handles energy input and conversion. In particular, the energy conversion follows simple energy balance equations, including thermal heat capacity of the station, and generates diffuse waste heat that is dissipated into the respective thermal zone in the room model. Energy demand for Oven and Freezer is controlled by a PI controller located inside the component. The energy model also interacts with the material flow by means of state events that indicate e.g. when the Oven has reached its target temperature and is ready to accept entities.

The building and energy system components on the other hand are mainly continuous models (as they do not directly interact with entities), except for simple control logics. The building model contains four thermal zones with homogeneous temperature distribution and calculates heat transfer across the walls between the zones as well as with the environment. For this, a temperature profile for the ambient temperature can be specified as simulation input, which allows to compare different weather conditions, e.g. summer vs. winter. A more sophisticated thermal building model has also been developed for the hyPDEVS simulator and is presented in [259]. This model is, however, not part of this case study.

Outside energy providers supply the energy across the system boundary to the energy system, which is billed to the customer using time-dependent energy pricing. Specifying energy price profiles allows to incorporate various energy cost effects that influence production planning. These energy price profiles are changed according to the respective scenario, see the following sections.

6.5.2 Scenario 1: One Week Planning Horizon

Table 6.2 presents two typical scenarios of demand lists (*Dplan*) needing to be scheduled over the course of one week (i.e. 168 h simulation time). The table lists different orders (coming from customers) with quantities and delivery due dates (measured from the start of the week). Duplicate entries constitute different orders to be delivered to different customers. For this scenario, the ambient temperature is kept constant with 20 °C. In addition, energy prices (see $c_s(t_i)$ in Equation (6.11)) follow the profile depicted in Figure 6.3, where, for testing purposes, the price for electricity $c_1(t)$ is reduced between 10 a.m. and 4 p.m. on selected days, while the gas price $c_2(t)$ is kept constant. The idea is that this accounts for cheaper energy from a photovoltaic system during sunshine. The base energy prices are based on real energy prices from Austria in 2018 and are taken from [95]: Electricity: 0.116 €/kWh. Gas: 0.04 €/kWh.

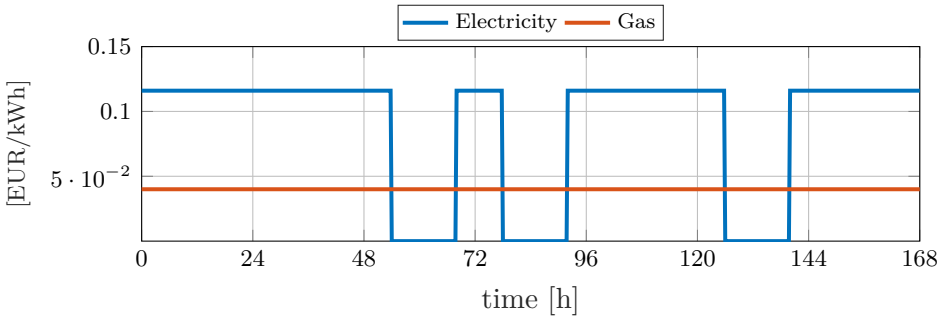


Figure 6.3: Energy price profile for Scenario 1. The price for electric energy is reduced between 10 a.m. and 4 p.m. on selected days to account for cheaper energy from a photovoltaic system during sunshine periods.

Scenario 1a is based on a real demand plan and features two different products (baked

Table 6.2: Demand plan for Scenario 1

Product Type	**Quantity**	**Due Time**	
		Scenario 1a	Scenario 1b
baked	6	48 h	168 h
baked	6	48 h	168 h
frozen	20	48 h	168 h
frozen	20	48 h	168 h
baked	24	72 h	168 h
frozen	50	72 h	168 h
baked	36	120 h	168 h
frozen	50	120 h	168 h
baked	8	144 h	168 h
baked	8	144 h	168 h
baked	8	144 h	168 h
frozen	25	144 h	168 h
frozen	25	144 h	168 h
baked	20	168 h	168 h
frozen	75	168 h	168 h

and frozen) in different batch sizes and with different due dates. Figure 6.4 shows the progression of the cost function over the iterations. The simulation time was 1 week, and the total number of VNS iterations was chosen as $n_{\text{iter}} = 100$. Further experiments showed that increasing the number of iterations does not improve the end result significantly anymore. Also, comparing repeated optimization runs (because of randomizations in the operators) delivered similar results. For the part goals, weighting factors were chosen as $\omega = (\omega_i)_{i=1\ldots5} = (1, 2, 0.5, 1, 1)$.

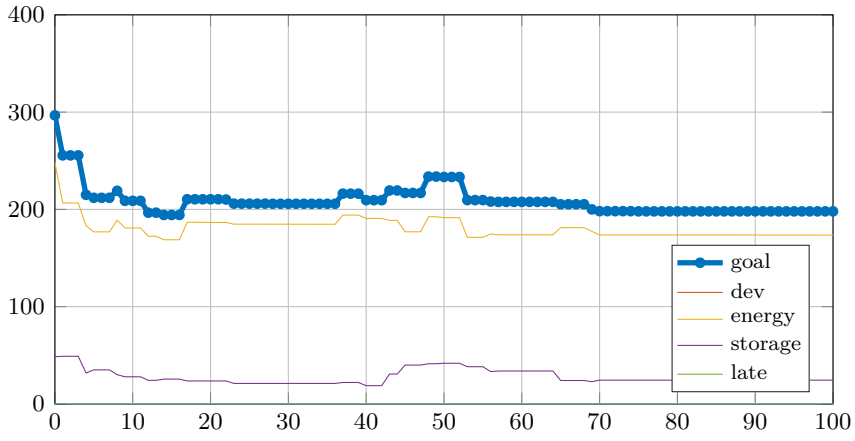


Figure 6.4: Progression of the cost function and part goals for Scenario 1a

The graph shows that the cost function also rises sporadically in between iterations, which is due to the SA acceptance criterion. However, the acceptance probability decreases with advancing iterations and the cost function decreases monotonically towards the end. When looking at the part goals, it shows that no significant constraints have been violated that would have caused penalties (entity deviation, delivery delay) and that the majority of the costs is divided between storage costs and energy costs. Overall, the cost function could be reduced by about 34%, with energy costs reduced by 30% and storage costs by almost 50%.

Figure 6.5 depicts in more detail the Oven and Freezer allocations (i.e. number of entities and temperature over time) in the initial solution, which was generated using the simple construction heuristic from Section 6.4.1, against the final optimization results. The different humps in the entity count correspond to different jobs.

The results show that the jobs are being grouped together around the due dates and that some are produced earlier than necessary, which saves on storage costs as well as energy costs, on the one hand by avoiding unnecessary thermal setup processes (i.e. reheating the Oven and cooling the Freezer), on the other hand also by optimizing the operating times. The gaps, where the stations are idle, increase, which allows to switch off the Oven and Freezer (shown in the figure) instead of continuing them to run, which would consume unnecessary energy.

This effect becomes even more apparent in Scenario 1b, where, compared to Scenario 1a, all delivery due dates are moved to the end of the week (cf. Table 6.2). This gives more leeway for job scheduling, but makes it more important to balance energy costs against storage costs. It also serves as a plausibility check for the optimization algorithm. The cost function for Scenario 1b is shown in Figure 6.6 with the final Oven and Freezer allocations presented in Figure 6.7. For the Oven allocation, it is noticeable that the jobs have been merged into two clusters and that, instead of pushing the two clusters together, a gap remains in order to take advantage of the reduced energy costs during the day (cf. Figure 6.3).

Intuitively, without considering energy costs, the best solution would be to produce everything as late as possible in order to minimize storage costs. This is also the solution depicted in Figure 6.8, where, for verification purposes, the same scenario has been tested with a modified cost function that excludes energy as a part goal. In particular, the coefficient ω_2 of f_{en} in Equation (6.7) was set to $\omega_2 = 0$. The respective cost function progression is shown in Figure 6.9. It is noticeable that a near-optimal solution is found after only a few iterations.

Table 6.3 compares the results of Scenario 1b with and without energy, by taking the respective final solutions and evaluating them using the same cost function from Equation (6.7), including energy (i.e $\omega_2 = 2$). This makes sense because, either way, the energy costs still have to be paid in the end, regardless of whether or not it is considered in the PPC. This comparison shows that incorporating energy costs together with other production goals during optimization can result in an overall better solution. While

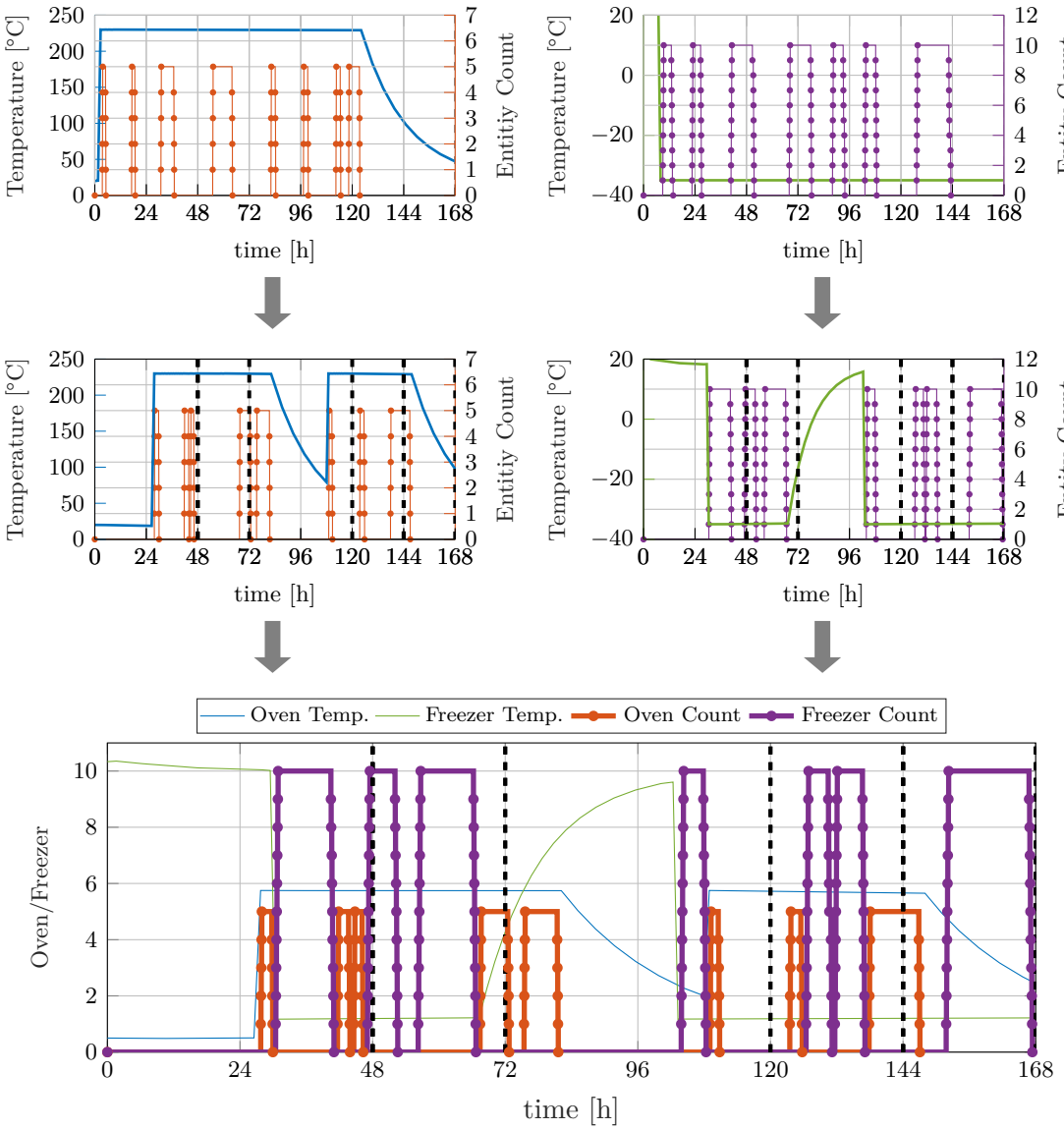


Figure 6.5: Optimization results for Scenario 1a, initial scheduling (top) and final results (middle), both for Oven (left) and Freezer (right) allocation, as well as combined view (bottom). Dashed vertical lines indicate job due times.

the storage costs have slightly increased, this is outweighed by the gain in energy costs. Not considering energy costs and only optimizing storage costs would ignore significant potential for cost savings.

While the runtime performance of the simulation-based optimization has not been studied in detail, the case study gives a first impression: The scenarios with $n_{\text{iter}} = 100$ iterations needed about 2600 - 3600 evaluations (i.e. simulation runs), with one simulation run taking

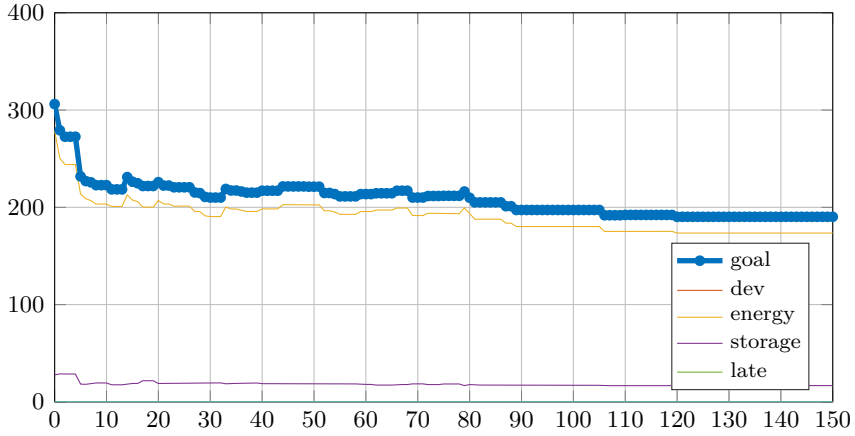


Figure 6.6: Progression of the cost function and part goals for Scenario 1b

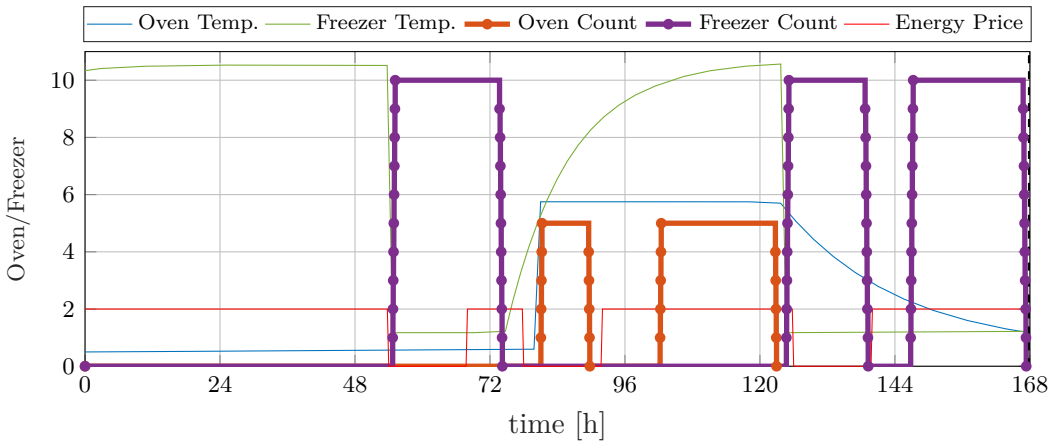


Figure 6.7: Oven and Freezer allocation for Scenario 1b

Table 6.3: Comparison between Scenario 1b with and without energy, showing final cost value using Equation (6.7) with $\omega_2 = 2$

	with Energy	**without Energy**	**Improvement**
Energy costs $\omega_2 \cdot f_{\text{en}}(x)$	173.53	188.92	8.15%
Storage costs $\omega_3 \cdot f_{\text{st}}(x)$	16.72	13.18	-26.86%
Total goal $f(x)$	**190.25**	**202.1**	**5.9%**

about 800 ms (1 week simulation time with 15 jobs). However, less than 30 iterations are often sufficient to find a near-optimal solution as Figure 6.4 and Figure 6.6 show. Overall, there is still room for decreasing the runtime by improving the implementation efficiency. However, these first results indicate feasible runtime also for larger scenarios.

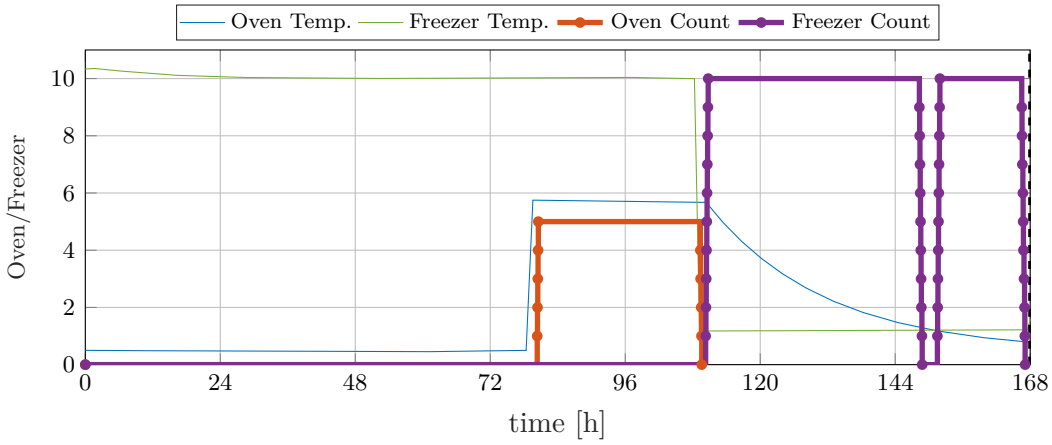


Figure 6.8: Oven and Freezer allocation for Scenario 1b without energy costs

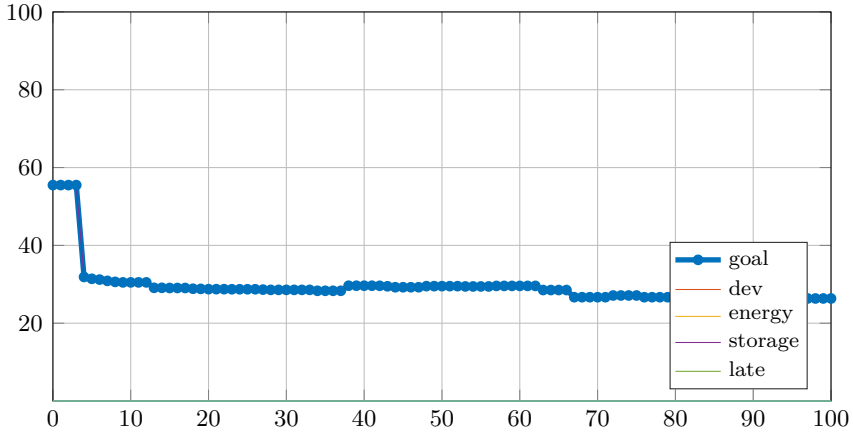


Figure 6.9: Cost function and part goals for Scenario 1b without energy costs

6.5.3 Scenario 2: Realistic Energy Price

In the second investigation, we look at a shortened scenario with a planning horizon of two days, which allows to better investigate the influence of energy costs and manually verify the planning results. Table 6.2 lists the Demand Plan (*Dplan*) for this scenario with quantities and delivery due dates. For the ambient temperature, we consider a real-world temperature curve presented in Figure 6.10 and we compare different energy price profiles $c_s(t)$, as shown in Figure 6.11. One electricity price profile $c_{1,\text{real}}(t)$ was taken from real historical electricity spot market data of the Austrian Energy Exchange (EXAA)[5], the other one $c_{1,\text{const}}(t)$ represents its mean value (0.0437 €/kWh) kept constant over the entire period. The gas price $c_2(t)$ is the same as in Scenario 1.

Scenario 2a presents a simple scheduling scenario with four jobs and two different delivery

[5]Source: https://www.exaa.at/en/marketdata/historical-data

Table 6.4: Demand plan for Scenario 2

Product Type	Quantity	Due Time	
		Scenario 2a	Scenario 2b
baked	32	48 h	48 h
baked	8	24 h	—
frozen	20	48 h	—
baked	4	48 h	—

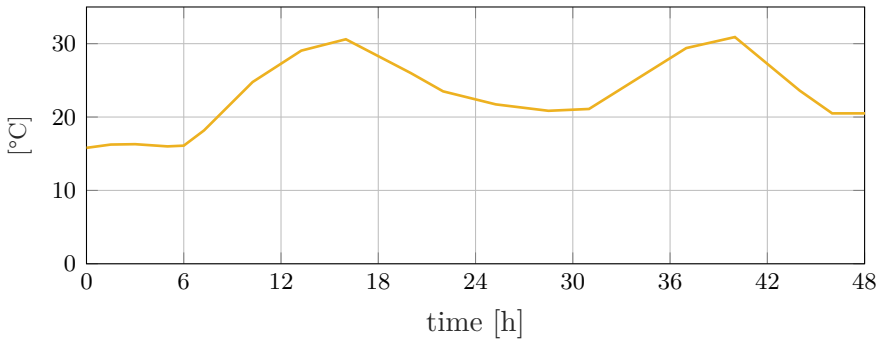


Figure 6.10: Ambient temperature curve used for the simulation, based on real data

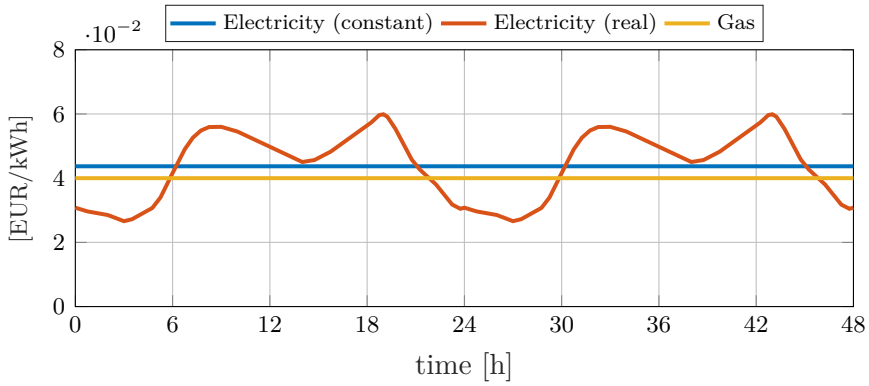


Figure 6.11: Energy price profiles, constant and real

due dates. Figure 6.12 depicts the Oven and Freezer allocations (i.e. number of entities and temperature over time) of the initial solution and the final optimization result when using the constant energy price profile. The optimization ran for $n_{\text{iter}} = 50$ iterations and the partgoal weights were chosen as $\omega = (\omega_i)_{i=1\ldots5} = (1, 2, 2, 1, 1)$, while the deterioration percentage of the SA acceptance criterion was lowered to $\Delta_{\text{SA}} = 0.3$.

The result is by and large as expected, with all jobs being produced as late as possible (due to the storage costs) and the one job (Job 2 in Table 6.4) adhering to the earlier delivery due date. While the overall goal was improved by 28%, not much energy was be

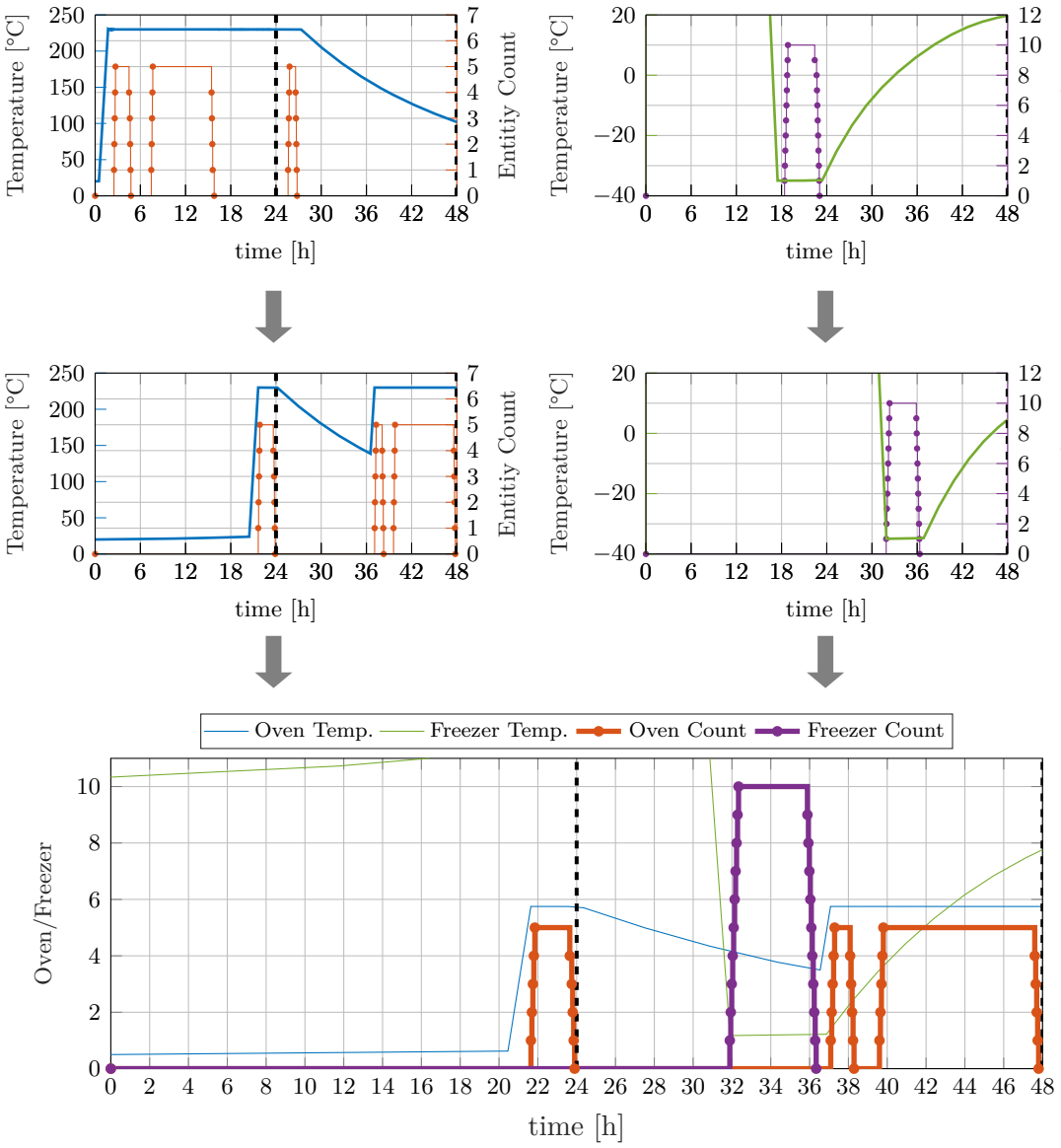


Figure 6.12: Optimization results for Scenario 2a, initial scheduling (top) and final results (middle), both for Oven (left) and Freezer (right) allocation, as well as combined view (bottom). The dashed vertical lines indicate job due times.

saved in this case, only about 6%, as there is not much leeway for shifting or merging jobs. This is also visible in the cost function plot in Figure 6.13. While the energy costs make up the majority of the target value, most of the improvements are achieved through the storage costs, which have been lowered by 77%.

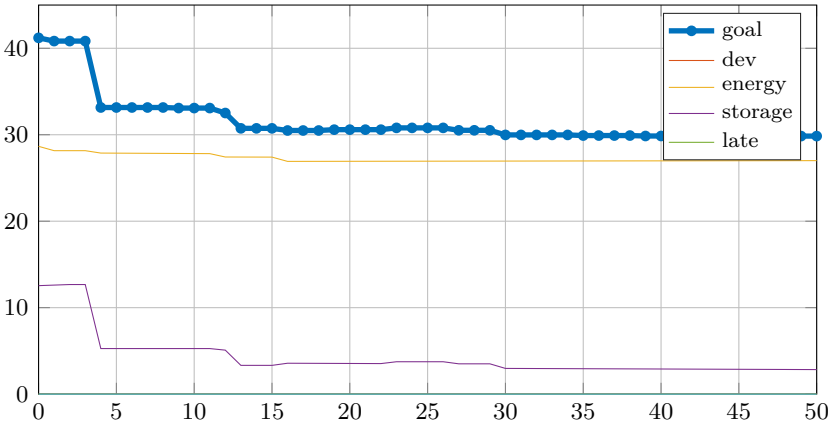


Figure 6.13: Cost function and part goals for Scenario 2a

Also, the variable ambient temperature itself does not have much influence on the optimization result either. As further analyses have shown, this is partly due to the temperature control in the building. On the one hand, this suppresses any direct influence of the ambient temperature on the thermal behavior of the production stations, on the other hand, a (more or less) constant total amount of energy has to be supplied to heat the building over the observation period, which is only reduced by the waste heat of the stations. However, it is irrelevant when exactly this waste heat is supplied.

Next, we want to take a deeper look at the influence the energy has on the planning result. For this, we omit the storage costs in the target system by setting the respective weight to $\omega_3 = 0$. This has the effect that the oven lots can be produced earlier, together with Job 2 that is due earlier, thereby saving a second setup process. Switching the sequence of the oven lots does not make much of a difference energetically. The result, which we will call x_c for further reference, is shown in Figure 6.14. Figure 6.15 presents the associated cost function.

If we compare this result with the planning result x_r obtained with the real energy price profile $c_{1,\text{real}}(t)$, shown in Figure 6.16, a different picture emerges. Here, the jobs are scheduled earlier, especially in the periods with low energy prices, while the periods of high energy prices are avoided. Especially the oven lots exploit these valleys, even though the oven continues to run during these gaps.

If we take x_c (Figure 6.14), which only considers a constant energy price, and evaluate it with the realistic variable price profile $c_{1,\text{real}}(t)$ (as it would be done in reality), we see that the goal would be about 5% worse compared to x_r, which considers the real energy price in the optimization, see Table 6.5.

The progression of the cost function, see Figure 6.17, shows that, compared to Figure 6.15, it takes more iterations to reach a near-optimal value, and the overall values are slightly higher. This will become even more apparent later in Figure 6.19.

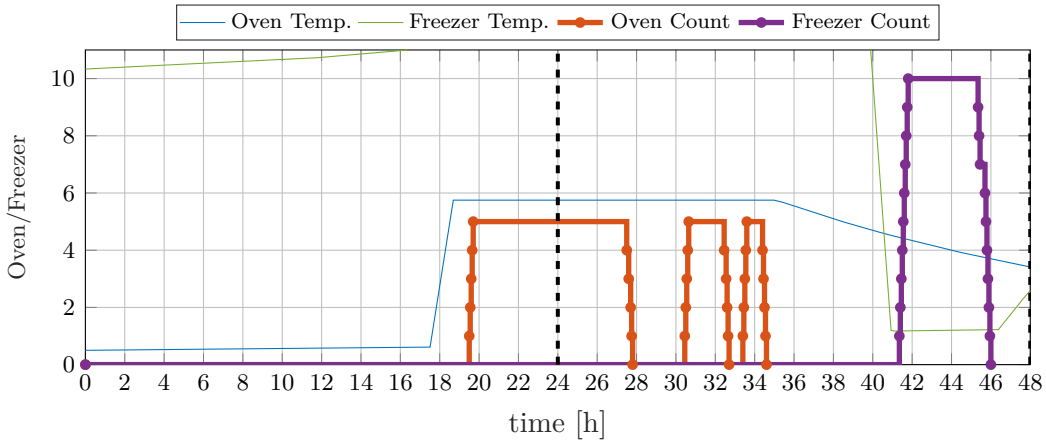


Figure 6.14: Oven and Freezer allocation x_c for Scenario 2a without storage costs and constant energy price

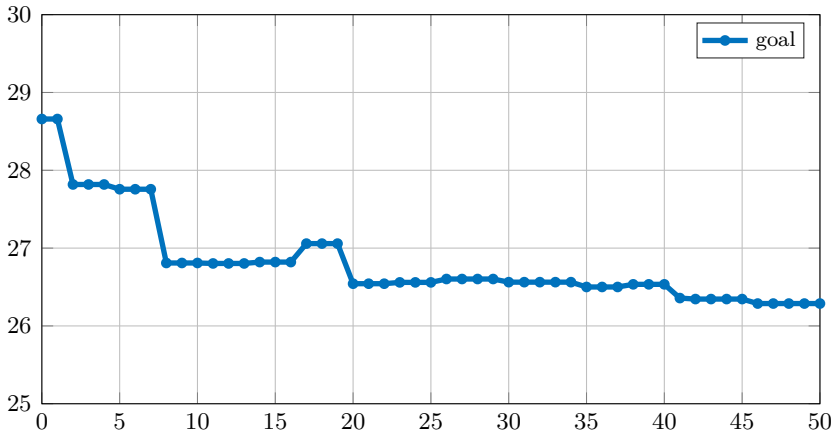


Figure 6.15: Cost function, i.e. energy costs, for Scenario 2a without storage costs

Table 6.5: Comparison of Scenario 2a between x_r and x_c, showing the cost value evaluated using the real energy price profile $c_{1,\text{real}}(t)$

	variable Price x_r	**constant Price** x_c	**Difference**
Goal $f(x)$	27.45	28.82	5%

The effect of job scheduling depending on the energy price can be examined more closely by looking at a single job as defined in Scenario 2b in Table 6.4. By sweeping a single job across the planning horizon, as illustrated in Figure 6.18, we can determine how the target function changes depending on the starting time x_1. The result is presented in Figure 6.19. Both with constant energy price and real price profile, the target function

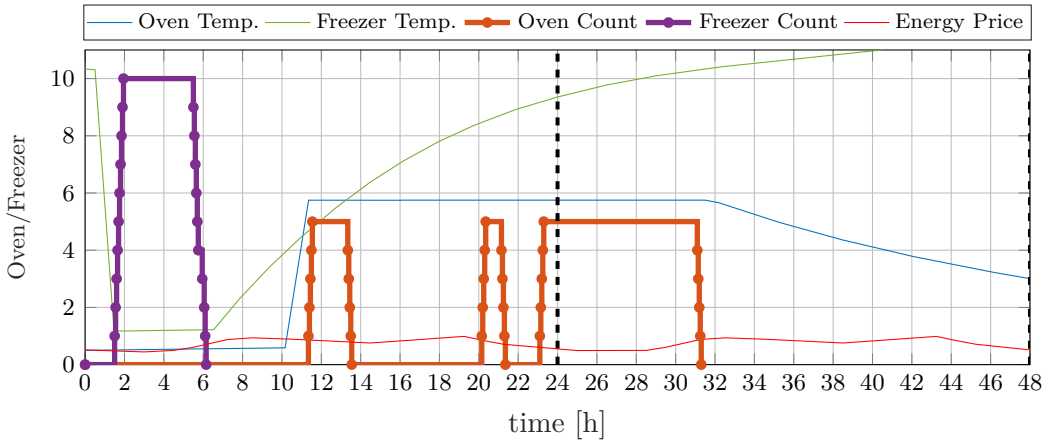


Figure 6.16: Oven and Freezer allocation x_r for Scenario 2a without storage costs and real energy price

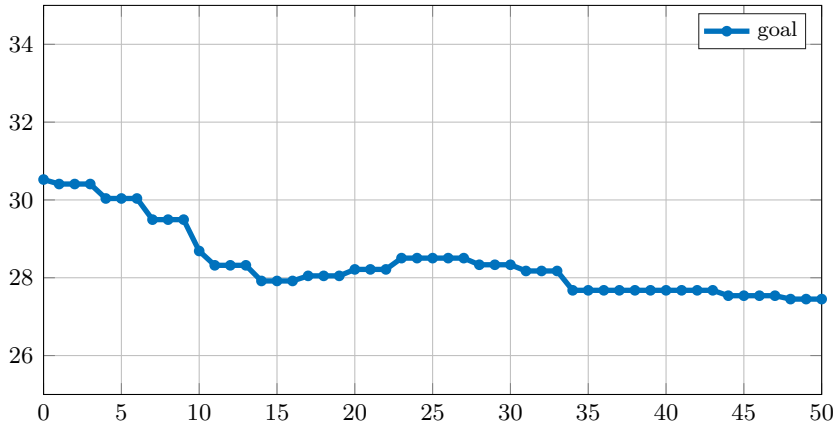


Figure 6.17: Cost function, i.e. energy costs, for Scenario 2a without storage costs and real energy price

has clearly pronounced hills and valleys that correlate approximately with the ambient temperature. However, the optimum values do not align, namely $x_1 = 27\,\mathrm{h}$ compared to $x_1 = 22\,\mathrm{h}$ with constant energy price.

What is also noticeable is that the target value is on average about 9 % higher when using the real energy profile, compared to the constant value, even though the mean price over the course of the day is the same, see Figure 6.11. These higher energy costs have to do with the fact that the energy demand is unevenly distributed throughout the day, not only due to production, but also, for example, due to intermittent filling of the heat and cold storage (that are part of the building TBS), which often occurs in times of above-average energy prices. Even if one could have suspected that this would balance out over the day, this does not seem to be the case here.

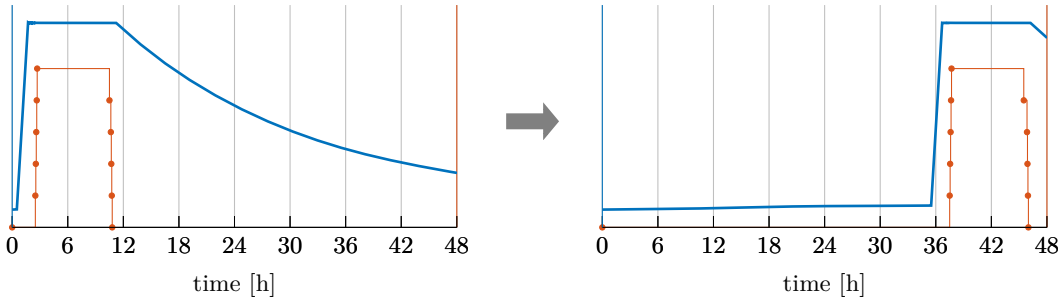


Figure 6.18: Sweeping the job starting time across the planning horizon, from 2 h (left) to 37 h (right)

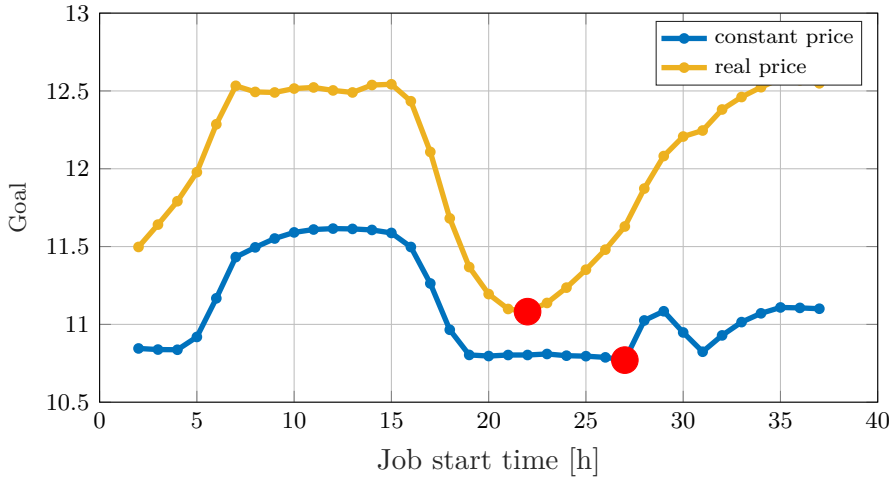


Figure 6.19: Target value plotted over the start time sweep, both for constant energy price and real energy price profile. The red dots indicate the respective minima.

6.6 Discourse

Results: The case study results show that including energy considerations into the planning optimization can potentially reduce overall costs and therefore provide better planning results. The case study has been simplified to highlight the essential characteristics, and the energy saving effects were recognizable but limited, which partly due to the model parameterization causing the building energy to make up a large part. However, it is easy to imagine a more complex production system with more diverse material flows, product types and higher energy turnover. This makes the optimization problem even more interesting because of the job completion times being more difficult to predict due to bottlenecks as well as even more potential for energy savings. From the current point of view, the presented simulation-based approach should also be applicable for these cases.

Many factors potentially influence the overall energy demand, which have to be included into the simulation. The building facilities contribute a significant portion to that, but also setup times of thermal processes can be optimized in order to save energy as well as improve production tardiness and reduce storage costs. These factors, however, have to be considered simultaneously in the target function, as part of a multi-objective optimization, instead of individually. Because of time-dependent energy prices, the actual energy costs for producing a certain product may vary depending on *when* it has been produced an can therefore not simply be calculated on an energy-per-piece basis.

Experiments on the case study have also shown that the Simulated Annealing (SA) acceptance criterion is important especially for the merging operator, since, although merging two jobs might be beneficial in the end, there might sill be a short-term increase in the cost function (e.g. if the products from the second job are being produced sooner and therefore increase storage costs). In the future, VNS parameter and neighborhood calibration might still be improved and even be automated as part of a hyper-heuristic approach [49] to directly adjust to the problem instance without the need for user intervention.

Simplifications: The optimization does not take into account economic aspects such as personnel costs, although these might potentially have an influence on the *Pplan* result. For example, electricity might be cheaper during the night, but personnel costs might be higher. However, such aspects can easily be included in the objective function as additional sub-goals or constraints (for example, as a shift schedule) without significantly changing the optimization method itself. Also, the simulation does not necessarily have to be extended for this purpose, since such economic aspects can also be evaluated outside the simulation. Also, availability restrictions like failure times or stochastic influences are not considered.

Energy Storage: It is important to ensure that any energy storages in the simulation do not distort the simulation result. For example, a part of the energy costs in one scenario may have been used to fill the storage tanks in the heating grid more than in another scenario. In order to take this into account, either the energy storage levels can be read out of the simulation and included in the target function evaluation, or alternatively warm-up and cool-down periods can be included in the simulation run to balance out the storage levels. We employed that latter approach, similar to other thermal simulation tools [285].

Constraints: The optimization approach does not require to formulate many explicit constraints as most of them are given implicitly in the simulation model. This makes the optimization easier to use, but the search less efficient. However, the GVNS operators implementing an improvement heuristic that ensures that only permitted moves are executed and thus the solution remaining mostly within the feasible range; the rest can be covered using soft constraints in the target function.

Flexibility: The target function can also be easily extended with additional optimization targets or constraints, such as lead time, capacity utilization or peak load penalties. In the same way, individual part goals can be deactivated by the user in case they are not relevant for the particular use case. The adjustable weights also allow the user to fine-tune the planning results to the individual needs. Furthermore, the GVNS method also lends itself to be extended by defining additional operators (or deactivate existing ones) without having to change the method. For example, a splitting operator could be useful to better optimize the batch sizes. All this makes the method very flexible and therefore suitable for a large number of applications.

One advantage of the GVNS meta-heuristic is that it allows to fine-tune the operators and neighborhood structures to the individual problem instance, thereby allowing more efficient search and decreasing computation time. Neighborhoods (i.e. switching, shifting, etc.) are explored repeatedly and iteratively instead of sequentially, and the trade-off between exploration and exploitation can be controlled by the user. In contrast to population-based approaches, like Genetic Algorithms, VNS as a single-solution-based methods needs fewer function evaluations (i.e. simulation runs), which is a topic of importance for simulation-based methods.

Comparison: For the discussed case study, an optimization based on a GA was also implemented as part of a research project, see [264]. A comparison shows a similar performance: While the optimization potential found is nearly the same, the GA typically requires significantly more evaluations, up to twice as many for a comparable scenario [261], due to its population-based nature. The GA is also more difficult to implement and must be heavily hybridized (e.g. targeted search as part of customized crossover and mutation operators), which is arguably more complex and less transparent with the GA than with the simple GVNS. On the other hand, the GA offers itself to easier parallelize the computations in order to decrease runtime, although there are also some parallelization strategies for VNS [73, 74]. For a more detailed discussion on the GA, we refer to [261, 264, 263, 252].

The hyPDEVS-based simulation itself, despite its hybrid nature, delivers sufficient performance to be feasible for simulation-based optimization tasks with with a large number of iterations. This has also been tested with larger real-world case studies, see e.g. [262].

CHAPTER 7

Framework for Model Engineering

After discussing the simulation and optimization methods that can be used in operative energy-aware production planning as part of a running PPC/APS system, we now want to look into an approach to support the engineering process of such systems for large-scale applications.

7.1 Introduction

In Chapter 4, we have already elaborated on the idea to manage the complexity of large-scale simulation models by applying abstraction and separation of concerns in the form of conceptual high-level modeling and modularization by means of Cubes. In the following, we formalize this workflow of transferring the conceptual description into a concrete implementation in the form of a modeling framework for developing hyPDEVS-based simulation models and optimizations using a Model-Driven Engineering (MDE) methodology.

It comes as an attempt to bridge the semantic gap between conceptual modeling and (hyPDEVS-based) simulation implementation by defining a domain-specific modeling layer in between. It provides a closer, more natural connection between the business-oriented concepts in which problems are usually expressed (i.e. the problem space) and the technology-oriented concepts in which solutions are described (i.e. the solution space) [45, 40]. The Cubes become first-class citizens in the development process rather than their implementation counterparts (i.e. Atomic and Coupled).

The model abstraction builds on the Cube concept and formalizes Cubes as a concept, including their interfaces, and the way they are instantiated, connected and configured. This ensures interoperability not just on the syntactic level, but on the semantic level as well [219], due to the domain-specific interpretation of Cubes. It also separates the high-level specification from the concrete implementation, which opens up a variety of

possibilities to derive not just hyPDEVS models, but also parameters necessary for the optimization procedure. It even allows to derive simulation models that might employ different modeling formalisms altogether, such as purely discrete models for simplified non-hybrid simulations [132], or even co-simulation implementations including coupling middleware.

While hyPDEVS is generic by nature, the new modeling layer adds domain-specific features on top of hyPDEVS, like interface types, in order to provide higher-level support for domain-specific models. It provides more intuitive modeling, with the hyPDEVS-based simulation execution still retaining the advantage of being formally sound.

The model engineering formalization builds on the findings from the research project BaMa and was only developed after. It is also not intended to replace the initial semi-formal workflow so much as provide additional development support for large projects. In that, it can be seen as an optional add-on on top of hyPDEVS modeling.

In the following section, we give a brief overview of the Model-Driven Engineering (MDE) methodology for developing software systems and discuss their applications in simulation engineering. We will clarify the different levels of abstractions and describe possible model transformations. After that, we describe the modeling framework for Cube-based application engineering and its implementation.

7.2 Model-driven Engineering Methodology

Model-Driven Engineering (MDE) is a methodology used in software engineering that provides a set of methods and guidelines to develop software systems using model formalizations and successive model transformations (i.e. manipulation operations on models) [56]. It therefore tries to provide a comprehensive vision for system development. The idea is that in MDE, models are the primary artifacts of the development process (as opposed to e.g. source code) and they represent the system and software at different levels of abstraction or detail [55]. The core concepts in MDE are modeling, meta-modeling and model transformations. All models (and that includes meta-models and transformations) are expressed in some notation, which MDE calls a modeling language. The MDE approach requires that the models and modeling languages are well-defined, since only a precise formalization allows unambiguous transformation and execution.

One popular statement often found in MDE is that "Everything is a model"[1] [40]. In particular, a definition of a modeling language itself can be seen as a model – this is referred to as meta-modeling (i.e. "modeling a model"). And this can be applied recursively: Modeling a meta-model means to define a meta-meta-model. More on meta-modeling is discussed in Section 7.2.3. Model transformations can also bee seen as

[1] As [40] points out, and quite an interesting analogy to keep in mind, is that MDE tries to make the point that the statement "Everything is a model" has a strong driving role in aiding the adoption of model-driven techniques in the same way that the principle "Everything is an object" was helpful in driving Object-Oriented Programming (OOP) techniques.

particular models (of operations upon models). Even the processes, development tools and resulting programs constitute models.

7.2.1 Overview

As models being the primary citizens in MDE, modeling is done at different levels of abstraction, a full-fledged MDE approach even leads to modeling the models themselves.

Figure 7.1 shows an overview of the main concepts considered in MDE. It involves two orthogonal dimensions: conceptualization and implementation. Implementation involves mapping the models to some running system and consists of three cores aspects: Model, realization (i.e. code in the case of software) and automation of the mapping in between. The conceptualization dimension is oriented to defining conceptual models for describing reality at three main levels: application (where models of the applications are defined, transformations are performed and actual executable code is generated), application domain (where the modeling language, transformation rules and implementation platforms for a specific domain are defined), and the meta-level (where conceptualization of models and transformations are defined) [40].

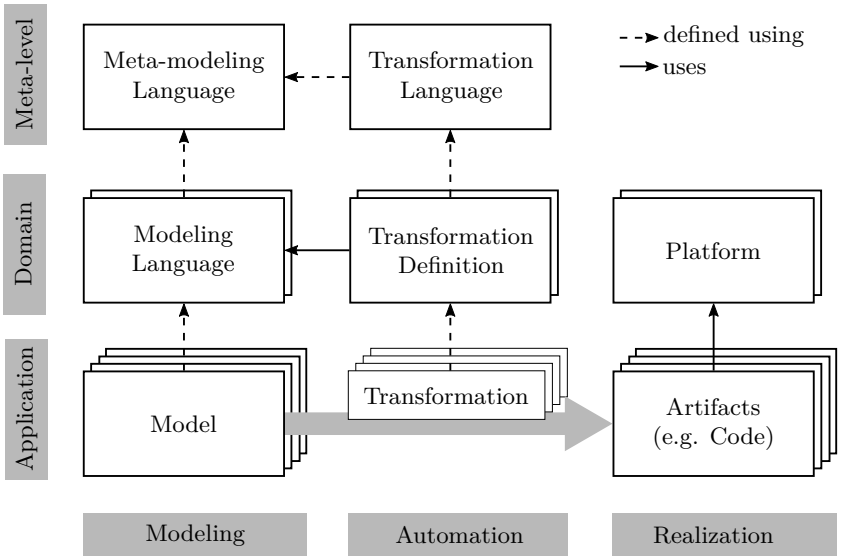


Figure 7.1: MDE methodology overview with two orthogonal dimensions: conceptualization (rows) and implementation (columns). Models are described using a modeling language and transformations generate executable artifacts from these models.

Models are specified according to a modeling language, which in turn is defined according to a meta-modeling language. One meta-modeling language allows to specify multiple different modeling languages, each of which in turn giving rise to a multitude of possible different models. Transformation executions are defined according to a set of transformation rules, which in turn is defined using a specific transformation language.

The core workflow in MDE for delivering executable artifacts (i.e. source code) spans from application model specification (left) to executable realizations (right) through subsequent model transformations with decreasing level of abstraction. This facilitates reuse of models and execution of systems on different platforms. The artifacts in our particular use case mainly involve executable simulation models, implying that they can also be called *models*.

Modeling languages are defined for a particular application domain and therefore define a model of said domain. The domain model is the conceptual model of the problem domain describing the various concepts, their attributes, relationships and their constraints and interactions. The purpose of formalizing domain models is to define a common understanding of the field of interest by defining its vocabulary and key concepts.

7.2.2 Modeling Languages

A modeling language provides a tool to specify models of a system. It is a means of expressing a concrete representation of a conceptual model in a formal and precise way by using some form of notation, such as diagrams, symbols, signs, letters, numerals, etc. In other words, a model is *defined using* (cf. Figure 7.1) a modeling language. This relation can be more precisely expressed as *conforms to*, i.e. "the model m *conforms to* the modeling language l". Model verification can check that this relation is correct [56].

A modeling language is defined by its abstract syntax, concrete syntax and semantics [9]. The abstract syntax defines the concepts and vocabulary provided by the modeling language as well as their relationships and well-formedness rules, i.e. how they can be connected to create models. An abstract syntax may be expressed using different concrete syntaxes, which provide a way to show the modeling elements in a concrete form to be worked and interacted with. So, an abstract syntax may for example have a textual concrete syntax (i.e. a textual programming language using keywords) as well as an equivalent graphical concrete syntax (i.e. diagrams depicting the abstract syntax in a concrete style and layout). The semantics of a modeling language explains the meaning behind the abstract syntax and is needed to allow a meaningful interpretation of the model.

For a particular application domain, Domain-Specific Languages (DSLs) can be designed to ease the task of describing things in that domain [40]. In case the language is aimed specifically at modeling, it may also be referred to as Domain-Specific Modeling Language (DSML). Prominent examples of Domain-Specific Languages (DSLs) that have been used in computer science for a long time are the HTML markup language for creating websites, VHDL for hardware description, SQL for databases and even Matrix Laboratory (MATLAB) can be considered a DSL for scientific computing. The counterpart to DSLs are General-Purpose Languages (GPLs), which are intended to be applied to any sector or domain. Unified Modeling Language (UML) or Java are considered to be General-Purpose Languages (GPLs). Although, the notion of *general-purpose* or *domain-specific* often

depends on the particular viewpoint (in particular what the considered "domain" is) and in some cases are up for discussion[2].

Models are developed in MDE using computer-aided tool support. Model editors implement a particular modeling language and allow to specify, view or change models interactively using a concrete syntax. Editors use a language parser to decompose a model according to the abstract syntax of its modeling language [199]. In most cases, model editors also provide extra features such as verification or syntax highlighting.

7.2.3 Meta-Modeling

As mentioned, models themselves can be seen as instances of some more abstract models. Hence, in the same way that a model is an abstraction of the real world, we can define a *meta-model*[3] as yet another abstraction that highlights the properties of the model itself [40].

Meta-models basically define a modeling language since they provide a way of describing a whole class of model that can be represented by that language. In particular, they define the abstract syntax of that language in terms of concepts, attributes, relationships as well as constraints present in a particular application domain. The language the meta-model describes is therefore a Domain-Specific Language (DSL).

Again, this idea can be applied recursively, giving rise to models describing meta-models, called meta-meta-models. While, theoretically, infinitely many levels of meta-modeling could be defined this way, software engineering in practice only uses three levels of abstraction with a common meta-meta-model that is defined on itself[4]. Figure 7.2 depicts these three modeling layers (M1, M2, M3), complemented by the layer M0 denoting the real-world objects that are being modeled (e.g. a real production plant).

The figure also gives an overview of the relationships between model, meta-model and modeling language. A model is said to be an *instance of* its respective meta-model, which in turn *represents* a modeling language, which the model itself *conforms to*. The relationship *conforms to* guarantees the validity of the model and is usually expressed implicitly via the *instance of* relation.

[2] For example, Petri Nets are considered by some to be a GPL [40], while one could also argue that they are *specific* for the domain of discrete-event dynamic systems. The same can be applied to DEVS. Even UML could be attributed to be specific for the domain of software engineering.

[3] The term *meta-model* means as much as "model of a model" and in our context refers to the meta-level modeling language. In the simulation domain, the meta-modeling term has previously been used in a different context, basically describing surrogate modeling (i.e. constructing simplified models that mimic the behavior of an underlying complex simulation), see e.g. [91, 62]. Since a surrogate model is in some sense a "model of a model" as well, the use of the term meta-model is perfectly justified, however it means something different.

[4] It might not appear intuitive how a model can be defined by itself. One can think of the example of a dictionary defining the words used in the English language. It does that by using the same words of that language.

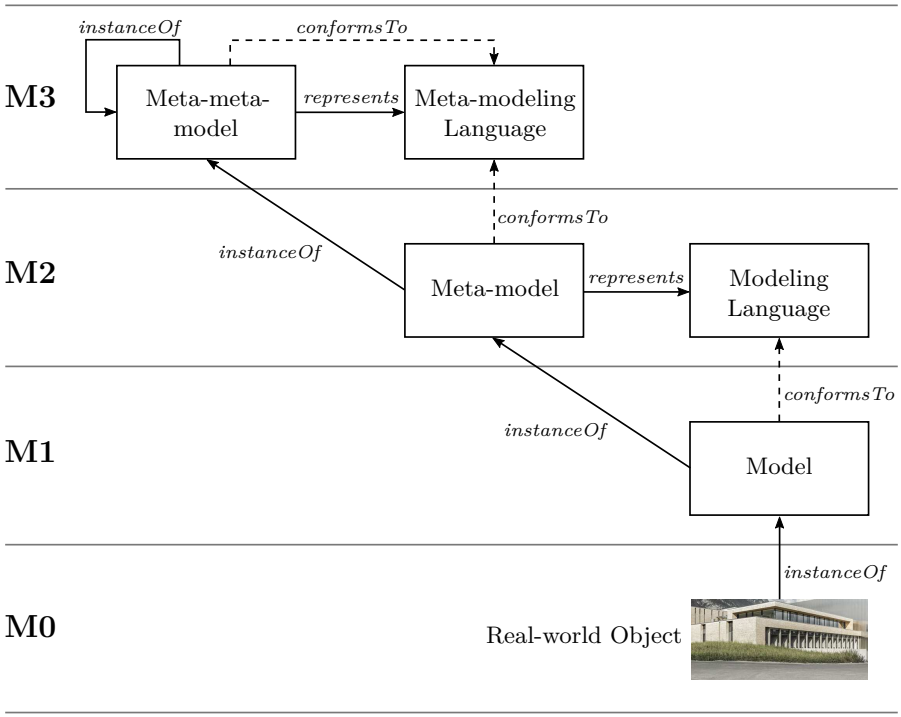


Figure 7.2: Four layers of modeling in MDE. Real-world objects are seen as instances of a model, which is an instance of a meta-mode, which is an instance of a meta-meta-model. Each model conforms to a modeling language, which is represented by a corresponding meta-level model.

This in analogous to the way a computer program conforms to the grammar of the programming language in which it is written. Such a grammar (e.g. for the language Java) is commonly defined using the Extended Backus-Naur Form (EBNF) notation, where EBNF itself constitutes the meta-meta-model.

7.2.4 Model Transformations

Model transformations (also called model morphisms [4]) allow to perform mappings between different models. Models can be transformed into another form according to a set of transformation rules. The goals of model transformations is to automatically generate different representations of a system in different views and abstraction levels. This way, instead of creating new models manually from scratch during different development stages, information that was modeled in one form can be preserved and reused in another form through an automated process.

An automated transformation requires that the models are specified in a well-defined modeling language and that the rules are formalized using a model transformation language. Transformation rules are defined at the meta-model level, and then applied at

the model level, namely on models that conform to these meta-models, see Figure 7.1 for reference. MDE solutions provide appropriate languages for defining transformation rules that are embedded in a meta-modeling environment where the transformation languages themselves also conform to a common meta-meta-model [56]. This is because transformation themselves can also be seen as models, and managed as such, including their meta-modeling [40].

Depending on the type of the output, model transformations can be categorized as Model to Model Transformation (M2M) and Model to Text Transformation (M2T) [292]. While a Model to Model Transformation (M2M) transformation converts a source model into a target model (which may conform to the same or a different meta-model), a Model to Text Transformation (M2T) transformation generates text from a source model, and is generally used for generating source code or supportive documentations. In the case of source code, M2T is also code model to code transformation, or code generation [56]. A more comprehensive overview of model transformation languages and methods can be found in [69, 142].

7.2.5 Model-Driven Architecture

Some of the principles in MDE are only described informally, there is no complete tooling support available for MDE yet. There are different specifications that describe the conceptual application of MDE principles, such as Model-Integrated Computing (MIC) [273], or the Model-Driven Architecture (MDA) [206]. The Model-Driven Architecture (MDA) is the particular vision of MDE in software design and development proposed by the Object Management Group (OMG) and relies on the use of other OMG standards [40]. Basically, it can be viewed as a specific MDE approach where the modeling and transformation languages are standardized by OMG. It is the most commonly used and accepted specification in the MDE practice.

MDA provides a set of guidelines for specifying and structuring models. It prescribes the use of meta-models and provides a common meta-meta-model, called the Meta-Object Facility (MOF). Using the Object Constraint Language (OCL) allows to specify constraints over meta-models. In particular, MDA introduces three types of models[5], classified according to decreasing level of abstraction: The Computation-Independent Model (CIM) focuses on a conceptual description of the environment and requirements. The Platform-Independent Model (PIM) describes the structure and operation of a system without specific implementation details. The Platform-Specific Model (PSM) combines the specifications in the Platform-Independent Model (PIM) with details on how the system is implemented on a particular platform [55]. The MDA also defines model transformations as the process of converting one of these models to another one. The intention is to have an initial model, then obtain intermediate models through successive model transformations before generating the final source code. This process greatly supports model continuity in MDE.

[5] In the view of the four-layer meta-modeling architecture, these models are located at the M1 level.

7.2.6 Model-Driven vs. Model-Based Terminology

It is worth elaborating in this context on the difference between *model-based* and *model-driven* engineering processes as well as related terminology. Both terms are used extensively in systems engineering and are sometimes used incorrectly.

Basically, *model-based* systems engineering [307] can be viewed as a process in which software models – in particular simulation models – play an important role. They are used e.g. for testing different design variants and operational scenarios or for designing controllers. The typical process here is that designers specify the domain models of the system, which are then handed out directly to programmers as blueprints to manually implement them. The term *model-driven* on the other hand implies a paradigm that uses models as the *primary* artifacts (i.e. they "drive" the process), where the software implementation can be (semi-)automatically generated from these models [40]. This in turn implies the use of model transformations and also meta-models (on which the transformations are defined). In that sense, all model-driven processes can be seen also as model-based, but not the other way around.

In the area of Model-Driven Engineering (MDE), other acronyms, such as MDA or MDD, are often found, some of which might appear to be synonymous at first glance. Their subtle differences might not be immediately obvious and so it happens that different uses can be found in the literature and that these terms are sometimes used interchangeably. We stick to the definitions given in [40]. Figure 7.3 gives a visual overview of the relationships between the most frequently used acronyms.

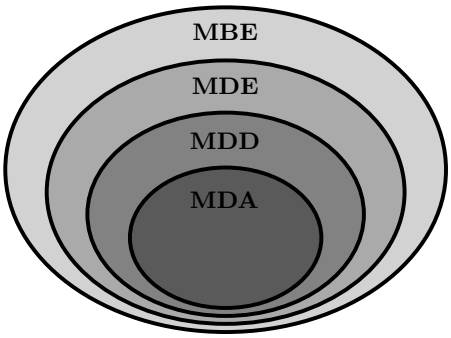


Figure 7.3: Relationships between the different acronyms in the MDE context

Model-Driven Development (MDD) refers to applying MDE as a paradigm for (software) development. It uses models as the primary artifact in the development process and derives implementations (semi-)automatically from them. In that sense, MDE can be seen as a wider term as *engineering* typically goes beyond pure development activities and might encompass other tasks as well (e.g. reverse engineering) [195]. Model-Driven Architecture (MDA) on the other hand is a particular vision of Model-Driven Development (MDD) proposed by the Object Management Group (OMG) (as described in Section 7.2.5) and thus relies on particular Object Management Group (OMG) standards [206]. In addition, there are more specialized application domains that have produced their own

corresponding terminology, such as Model-Driven Software Engineering (MDSE), Model-Driven Product Engineering (MDPE) (both sub-sets of MDE) or Model-Based Systems Engineering (MBSE) (as a sub-set of MBE) [40].

In the following section, we describe a model engineering framework that we have devised for developing Cube-based application models of hyPDEVS simulations.

7.3 Modeling Framework for Cube-Based Application Engineering

In an effort to improve the ease of use for application engineers (which typically are domain experts, but not hyPDEVS experts), we formalize a simplified abstraction from hyPDEVS as a platform-independent and domain-specific modeling layer [133, 132, 21]. This modeling abstraction allows engineers without in-depth knowledge of Modeling and Simulation (M&S) to engineer models for specific applications by instantiation, parametrization and configuration of pre-developed Cube components. From the abstract specification, executable implementations can be derived to be deployed as part of PPC/APS planning modules in practice.

The conceptual framework follows the MDE paradigm and formalizes the steps outlined in Chapter 4. Instead of informal model descriptions, we introduce an abstract modeling layer that allows to formalizes component-based application models using Cubes in a high-level manner by abstracting away unnecessary implementation details.

In its general idea, the envisioned model-driven application development workflow encompasses three different levels of abstraction, shown in Figure 7.4. The levels are based on the ones described by the Model-Driven Architecture (MDA). Starting from a first (informal) Conceptual Model (CM) (developed by the application engineer together with the problem owner), application engineers formalize a platform and simulation system-independent model description (PIM) in a domain-specific modeling environment using a Domain-Specific Language (DSL) by instantiating and configuring Cube components. The PIM is intended to be independent of specific implementation details in order to facilitate reusability of the higher-level specification for different implementations. The developed PIM can then be transformed into different platform and simulation system-specific model implementations (PSM) to generate executable source code using automated model transformations (as described in Section 7.3.3) that enrich the high-level specification with extra knowledge about implementation-specific details to obtain the full model.

This introduces a platform-independent modeling layer that raises the level of abstraction for the user to develop application models. Aside from more intuitive and efficient modeling by using domain-specific features and components, one of the main benefits of the PIM is that it separates model specification from the concrete implementation (again, following separation of concerns), thereby giving rise to the ability to derive multiple different executable simulations from the same abstract specification [58]. This

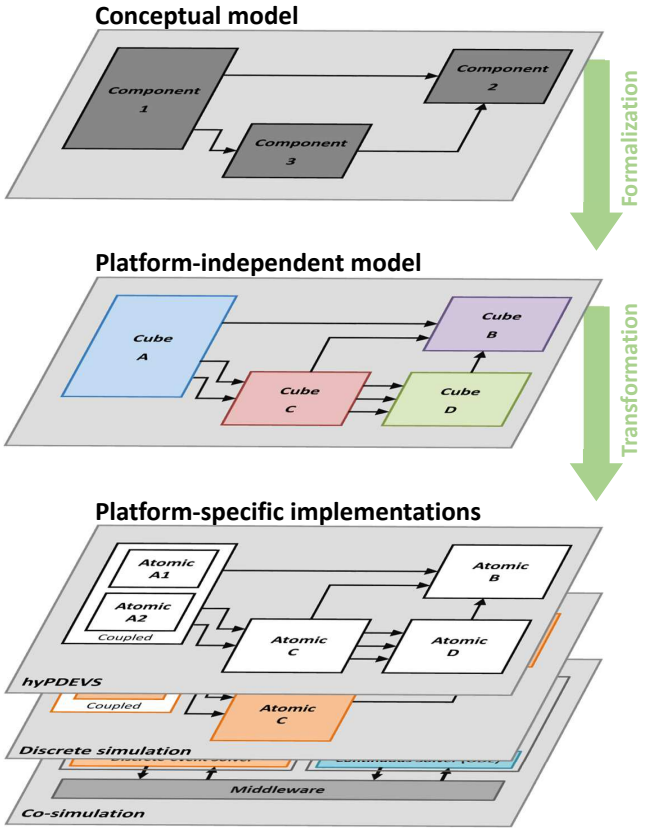


Figure 7.4: Model development process with decreasing levels of abstraction: Conceptual Model (CM), Platform-Independent Model (PIM) and different Platform-Specific Model (PSM) implementations

includes not just hyPDEVS simulations, but potentially others as well that might employ different modeling formalisms, such as purely discrete models for simplified non-hybrid simulations [132], or even co-simulation implementations including coupling middleware. The applications are not bound anymore to hyPDEVS, but instead may benefit from (or be combined with) other simulation methods. The PIM serves as a way to manage component instantiations, parameterizations and coupling configurations.

We focus here on developing component-based system models by instantiating, coupling and parametrization of predefined Cube model components. The internals of the Cube models are implemented in a Cube library.

The PIM modeling layer and corresponding DSL are defined by means of a meta-model, called hyPIM. This meta-model contains Cube templates that reference the respective implementations in the Cube library. For the M2M transformation specifications, an additional meta-model for the Platform-Specific Model (PSM) is introduced that represents the grammar of the simulation formalism, i.e. hyPDEVS. M2T code generation is

used to obtain an executable artifact representing the planning module. The hyPDEVS simulation thereby uses the Cube implementations from the Cube library. Technically, the resulting source code confirms to a meta-model as well, namely the one specified by the grammar of the respective programming language (cf. the explanations in Section 7.2.3). Figure 7.5 gives an overview of the complete modeling framework.

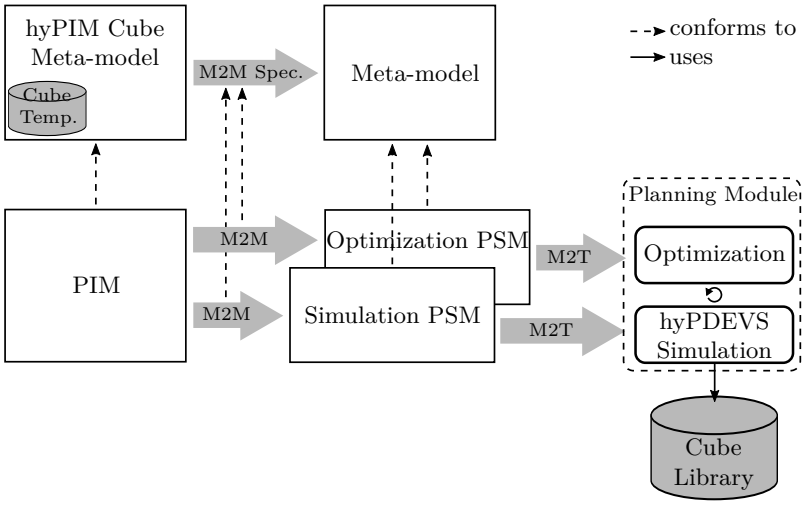


Figure 7.5: hyPIM modeling framework with PIM, PSM as well as respective meta-models, M2M transformation and M2T code generation

The PIM model specification includes not only the information relevant for deriving the simulation, but the optimization as well (in terms of a second PSM) in order to obtain a complete APS planning module ready to be deployed in industry. Some parts within the optimization module involving not the meta-heuristic per se, but rather the interfaces to the simulation depend on the particular application model, in particular for the target function and *Pplan* generation. Some of these application-dependent aspects are redundant to the simulation – the PIM modeling helps to avoid inconsistencies and therefore aid in more efficient development.

7.3.1 hyPIM Metamodel

As mentioned, the platform-independent modeling layer is formalized by providing a formal meta-model, to which all application models have to conform. The meta-model also forms the abstract syntax of a domain-specific modeling language [40].

PIMs are specified as an instance of the Cube meta-model, called hyPIM. Figure 7.6 shows the basic elements of the meta-model. It formalizes how platform-independent models can be specified from pre-defined Cubes. Some details of the meta-model have been omitted for reasons of clarity. Each model (class *System*) is composed of *Cube* instances, which are separated into four domains: *BuildingCube*, *EnergyCube*, *LogisticsCube*, and *ProductionCube*. Each of these domains defines certain Cube components (templates, to

be exact), such as the *Oven* (depicted), *BuildingHull*, and others like *Splitter*, *Combiner* or *Chiller*. Cubes can be arranged hierarchically, i.e. a Cube can contain other Cubes. A Cube instance also comprises ports (*EntityPorts*, *EnergyPorts*, and *InformationPorts*) as well as *Connections*. Also, each Cube may incorporate a list of parameters, including *Pplan* and *Aplan* [133].

Distinguishing different types of ports according to their respective domain has the advantage that, during model development, the model editor can check the validity of the connections (i.e. that both ports of a connection have the same type) at design time, thus supporting the application engineer to reduce errors in the model. The meta-model incorporates Cube templates that reference concrete Cube implementations that are given in the Cube library presented in Chapter 5. This is further discussed in Section 7.3.4.

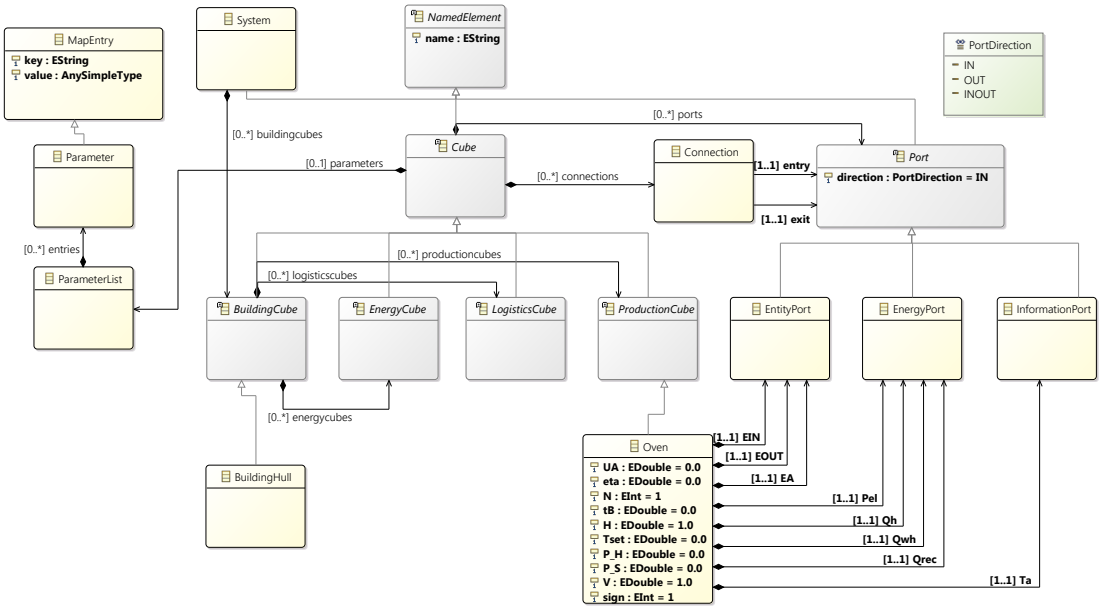


Figure 7.6: Overview of the basic hyPIM meta-model

7.3.2 hyPDEVS Metamodel

For transforming the PIM into a hyPDEVS-specific PSM implementation using M2M transformation, we also need to formalize a meta-model for the hyPDEVS formalism. Several DEVS meta-models have been proposed in the literature [57, 203, 240, 315]. Since most of them focus on Classic DEVS, we had to adapt these meta-models for hyPDEVS, shown in Figure 7.7.

The meta-model formalizes not just hyPDEVS couplings, but the internal Atomic functions (δ_{ext}, δ_{int}, λ^d) in order to capture the complete formalism as a standalone meta-model that can be employed for other purposes in other frameworks as well.

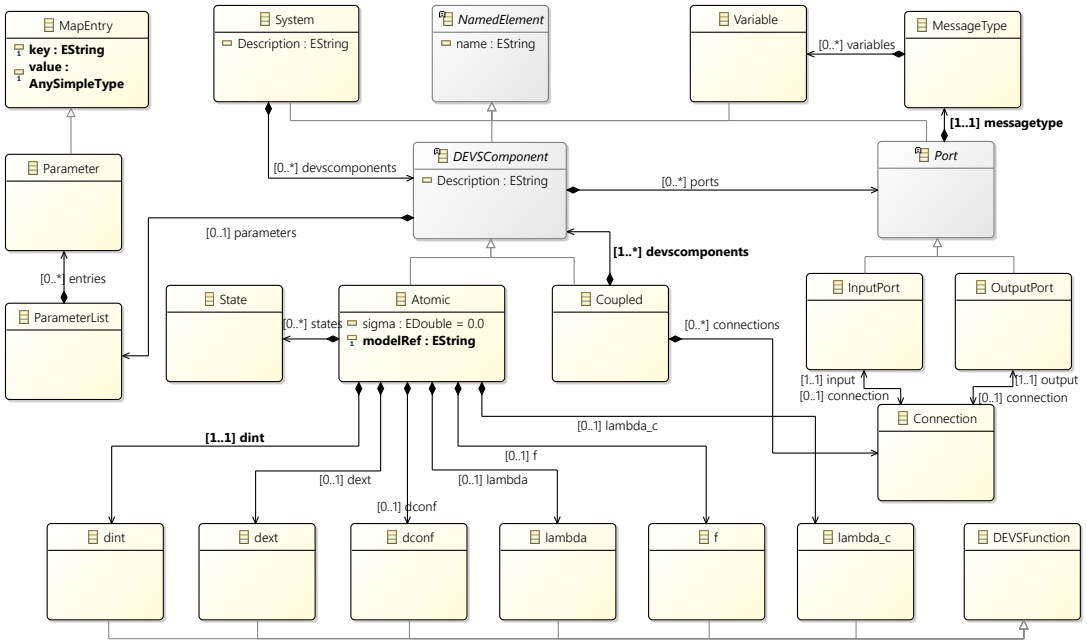


Figure 7.7: Overview of the hyPDEVS meta-model

7.3.3 Model Transformations

A Model to Model Transformation (M2M) transformation between the PIM and PSM is specified using the respective meta-models. For that, all meta-models need to conform to a common meta-meta-model. In this stage, the PSM can be further refined by a M&S expert to add additional knowledge for the simulation execution. Subsequent M2T transformations generate executable source code from the PSM. Thereby, information about parameters and couplings is supplemented with source code describing the internal dynamics, which is implemented in custom PSM component libraries. More details on the transformations and their implementation are given in Section 7.4.

7.3.4 Cube Library

For including simulation model component libraries as part of a component-based paradigm, model templates can be employed, which, during model transformation, are converted into concrete component implementations [55]. By using model templates, formal PIM and PSM models can be developed. However, these model templates need to have an associated concept in the PIM layer that the user can instantiate within the DSL editor. In other words, this concepts needs to be part of the PIM meta-model. This might be fine for a domain-specific library that in itself is complete (for example, [55] presents such a meta-model for the BPMN formalism). If, however, the library needs to be extensible (and in fact extending component libraries is often a significant part in the

modeling workflow), this would require frequently changing and updating the respective meta-model, which should be avoided in order not to break existing application models.

While one can think of other approaches of integrating component libraries into the PIM modeling layer, the problem does not seem to be trivial. In the four-layer meta-modeling architecture (M0-M3, see Figure 7.2), component libraries are located at the M1 layer since they are essentially models and have to conform to the same meta-model in order to be able to be composed into application models that are valid with respect to this meta-model. So, in order to enable instantiation of components, this would require a different instantiation relation within the M1 layer, in addition to the one between M1 and M2. However, such functionality is currently not provided within classic MDE solutions like MDA.

It appears that the problem is more fundamentally rooted in the four-layer meta-modeling architecture and a comprehensive and consistent solution requires deeper research into multi-level meta-modeling. Atkinson and Kühne [176, 15, 10, 175] discuss a multi-level meta-modeling approach that is not bound to the traditional M0-M3 layers. They discuss use cases that require multi-level meta-modeling, where modeling in the M0-M3 architecture would result in "accidental complexity" [11]. This is due to the fact that the definition of the problem domain in this case is restricted to one meta-level (M2), using typical meta-modeling mechanisms like type definition, inheritance and cardinalities [200, 14, 12]. If such features are needed on the M1 level, they would have to be modeled explicitly on the M2 level [180]. An example of such a use case would be the *type-object* pattern, where the user needs to be able to define types (i.e. classes) as well as create instances (i.e. objects) from these types. Several approaches and workarounds have been developed over the years to deal with these restrictions [13, 200].

In an effort to allow a more natural way of expressing multiple levels of logical classification, the authors introduce a concept of *ontological* instantiation within the M1 layer, in addition to the classic *instanceOf* relation between M1 and M2, which in this context is referred to as *linguistic* instantiation [9]. This added ontological dimension allows to define simulation model component libraries, whose instances can all be further instantiated, making them ontological meta-types [200]. This allows for multi-level (deep) instantiation, essentially blurring the boundaries between class and object concepts. In [7], the authors introduce the term Clabjects to express elements with both type and instance aspects.

A similar technique can also be found in modeling languages such as Modelica, where each model is automatically raised to be a class that can be instantiated and parameterized again in a larger context [196]. It becomes evident that such a multi-level approach would make sense for introducing component libraries in an MDE workflow. However, there is currently very little tool support for multi-level methods in MDE as the community is more focused on domain modeling rather than language engineering [8, 200]. Further research efforts are required to facilitate wide-spread adoption.

7.4 Implementation

The presented meta-models allow to specify a Model to Model Transformation (M2M) transformation from the PIM to the PSM as well as generate executable source code from the PSM using Model to Text Transformation (M2T) transformation, thereby formalizing the model development workflow described in Chapter 4 and depicted in Figure 7.4.

The framework was implemented as a proof of concept to demonstrate its feasibility for developing hybrid Cube-based application models. As a concrete case study example, we chose the Simple Production Line described in Section 5.7 in the MatlabDEVS prototype implementation.

The meta-models have been implemented using the Eclipse EMF framework [268] and Xtext [25]. Each meta-model itself is an instance of the Ecore meta-meta-model provided by the EMF. Figure 7.8 gives a more detailed overview of the implementation, which has also been published in [132, 133].

Additional constraints on the meta-model are defined using the OCL language [296]. Such constraints allow for example to check whether or not all connections are valid and parameter values are within allowed ranges.

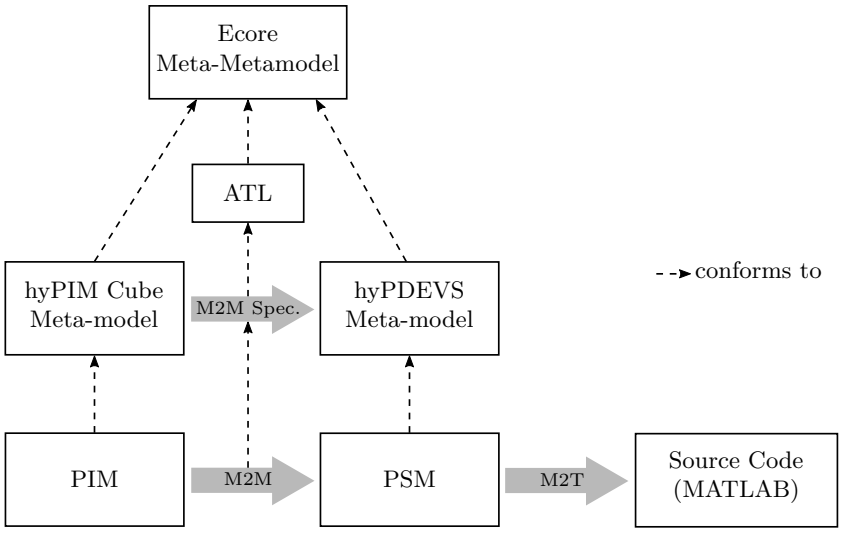


Figure 7.8: Implementation of the modeling framework in Eclipse EMF. The meta-models conform to the common Ecore meta-meta-model provided by the EMF. The M2M transformation is implemented using ATL and the M2T transformations are based on Xtend.

7.4.1 Model to Model Transformation

For producing a hyPDEVS target model from a hyPIM source model using M2M transformation, Atlas Transformation Language (ATL) provides a declarative way of specifying transformation rules between the respective meta-models [40]. On the M1 model level, a transformation engine executes these rules to produce the output models.

The Atlas Transformation Language (ATL) transformation may produce a single or multiple *DEVSComponents* (*Atomic* or *Coupled*) from a single Cube. Also, ports and connections can be instantiated and modified accordingly. In particular, for each entity connection (E_{out}–E_{in}) given in the PIM, an additional feedback connection (E_{incom}–E_{outcom}) is being added as an acknowledgment channel, see Figure 7.9.

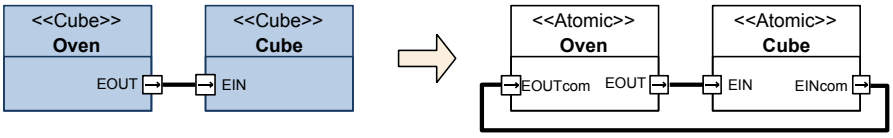


Figure 7.9: An entity connection in the PIM (left) is transformed into two connections (with respective ports) to include the feedback connection (E_{incom}–E_{outcom}) [133]

The feedback connection can be derived automatically from the information given in the PIM. It is therefore not necessary to model it explicitly in the PIM, thereby making model development easier and more efficient for the application engineer. The necessity of the feedback connection in the hyPDEVS implementation has been discussed in Section 5.5.4.

7.4.2 Code Generation

M2T transformations generate MATLAB source code from the hyPDEVS PSM to be executed using the MatlabDEVS toolbox (see Section 5.6 in Chapter 5). The M2T transformations are realized by means of the Xtend language [40] within the EMF using code templates, which are populated with dynamic data from the PSM to instantiate, parametrize and compose the Cube models directly form the Cube component library which is implemented in MatlabDEVS. The resulting source code forms an executable application model, which can then be deployed and simulated using the simulation engine provided by the MatlabDEVS toolbox.

7.5 Discourse

Engineering Process: The most notable advantages of employing an MDE approach are rapid model development and increased productivity due to an increase in efficiency. Although it may take a large amount of time to provide the initial infrastructure and develop the necessary meta-models, however, once these are developed, then subsequent development time and costs decrease significantly. MDE is different from traditional development approaches in that it has a steeper learning curve and requires to change

established programming habits. However, employing meta-modeling shows its advantages once it is understood well. Although traditional model engineering is beneficial for small-scale projects, the model-driven approach improves management of large-scale models [55]. This is also due to the model transformations reducing (if not solving) the model continuity problem (see Section 4.2.2).

Using Component Libraries: One of the main issues we have encountered concerns the use of simulation model component libraries. While it is certainly possible in principal to incorporate the use of component libraries into an MDE framework using model templates and workarounds, as described for example in [200, 55], the particular Eclipse Modeling Framework (EMF) environment used in our implementation is still lacking native support that would accommodate such an approach naturally, ideally by allowing for ontological instantiation. Considering that Eclipse EMF is viewed by many as the de-facto standard in model-driven software engineering, it becomes evident that there is still research to be done in this regard.

Reference Model: When looking back to Figure 3.3 from Chapter 3, it becomes apparent that the reference model there has a similar idea behind it as the hyPIM meta-model. In that sense, the reference model represents some kind of meta-model, although being less formal.

CHAPTER 8

Conclusion

8.1 Discussion

In this work, we have investigated methods for model engineering and simulation-based optimization in energy-aware production planning. The optimization results in the case study show that incorporating energy considerations into the planning optimization can potentially reduce overall costs and therefore provide better planning results. Although the case studies have been simplified to highlight the essential characteristics, it is easy to imaging applying the described methods to more complex production systems [129]. The hyPDEVS-based simulation, despite its hybrid nature, delivers sufficient performance to be feasible for simulation-based optimization tasks with a large number of iterations.

We have discussed a concept for model-driven development of hybrid simulation models for interdisciplinary simulation of production systems. A platform-independent model specification, separated from implementation details, allows practitioners to engineer simulation-based planning solutions for new application cases. It aims to provide a way for non-expert application engineers to easily develop component-based application models.

The focus of this work was on developing the methods necessary to facilitate energy-aware production planning rather than presenting case studies themselves. This is why we do not provide real-world parameterization or discuss quantitative results. However, the fact that the case studies are still derived from real production plants ensures the applicability of the methods and relevance for real-life applications.

We have already discussed some of the issues regarding hybrid simulation, meta-heuristic optimization and model engineering in the respective chapters. In this chapter, we aim to provide an overall evaluation and to answer the research hypotheses that were proposed in the introduction.

Hypothesis 1: *A simulation approach based on DEVS is feasible for developing hybrid simulation models of real-world production systems that are modular, reusable and facilitate separation of concerns.*

Modeling based on hyPDEVS allows to capture discrete and continuous aspects of production components in an integrated and modular manner. This is an important feature for comprehensive and interdisciplinary analysis of energy efficiency in production systems and the main advantage of this approach. The fully integrated dynamic simulation enables to investigate dynamic interdependencies between material and energy flows. The formal approach is very generic and makes little restrictions on what can be modeled in principle. As one of the main advantages, hyPDEVS greatly facilitates modularity of hybrid components and hierarchical model composition in a formally sound manner, due to its proven closure under coupling. Thus, the Cube concept can be implemented well using hyPDEVS. We were able to develop simulation models of different case studies by reusing pre-developed Cube models. Some of the components have been reused multiple times, even in different configurations (e.g. the Oven Cube used as a Freezer). This demonstrates the practical applicability of this method. A standalone simulator prototype was developed that uses a hyPDEVS simulation engine and is now part of a PPC/APS software solution of one of our project partners, ready to be applied to real-world applications.

However, as there is not domain-specific tool support available yet, M&S experts have to develop component libraries from the ground up. This initially requires substantial development effort before the simulation tool is ready to be deployed in practice. But, as the library grows, this should become easier and subsequent development effort should decrease. Due to the generic nature of the formalism, the modelers have to address basic simulation aspects, such as entity acknowledgment. A sound and comprehensive approach is needed to incorporate these aspects into entire libraries of model components in a consistent manner and to ensure interoperability [229]. For domain-specific hyPDEVS-based modeling in practice, it is a good idea to develop domain-specific specializations, e.g. on the implementation level using modern software engineering techniques. Another way to provide domain-specific features it to introduce a modeling layer on top of hyPDEVS, as we have described in Chapter 7. Such features allow to specify domain-specific models more efficiently and abstract away specific peculiarities. A preliminary investigation on this topic was also conducted in [130].

Yet, it might still seem excessive to develop simulation libraries from scratch instead of building on established simulation tools and couple them together for a hybrid co-simulation. So, why not use co-simulation? It depends on the intended application. While co-simulation certainly has its advantages for certain use cases, in particular less implementation effort for single-use simulations, in which case co-simulation is clearly to be preferred, hyPDEVS-based simulation on the other hand is better suited in cases where requirements such as reusability, trustworthiness or performance are more important. From a practical standpoint, in cases where the simulation is intended to be part of a PPC solution to be deployed in the field, it is in most cases not practical to employ

commercial simulation tools that would require software licenses. This requirement alone rules out many off-the-shelf tools (e.g. MATLAB/Simulink/SimEvents).

Hypothesis 2: *The integration of material and energy flow simulation enables meta-heuristic optimization techniques to provide energy-aware production planning.*

The hybrid discrete/continuous simulation enables to assess material and energy flow dynamically overtime as well as across different engineering domains in an integrated way, including dynamic interactions. This way, different production scenarios can be evaluated and compared during optimization, both in terms of energy demand and the influence of energy on other production variables (e.g. setup time and throughput). Especially for energy-intensive thermal processes, transient effects (e.g. heating up an oven) can be incorporated with sufficient level of detail to provide accurate measures of their energy consumption over time.

Furthermore, the simulation approach is not just limited to the domains considered so far, but may be applied to other areas as well as needed. For example, it would be easy to incorporate different types of entities into the simulation that depict e.g. mobile production equipment, whose limited might be factored into the optimization as a planning constraint. It has been demonstrated that additional constraints may very easily be included into the target functions as a soft constraint without affecting the general meta-heuristic.

The integration of energy efficiency aspects into modern PPC solutions allows to decrease not just energy costs, but also energy demand directly. The planning algorithm might for example shift oven jobs such that the produced waste heat benefits heating and cooling demand in the building. This increases the overall energy efficiency in production, and is also good for the environment. Application of the developed optimization method on a case study of an industrial bakery demonstrates the practicability for real-world applications. In a more comprehensive case study of that same bakery, which is presented in [261], it was demonstrated that, by means of hybrid simulation and meta-heuristic optimization, the required energy input can be reduced up to 30% in real production scenarios.

A necessary prerequisite – and here lie the limitations of energy-aware production planning – is that there is enough leeway available in production scheduling to save energy costs without influencing other production targets (e.g. delivery tardiness) too negatively. But in the end, it must be left to the operator to decide how to weigh the different part goals against one another to achieve balanced manufacturing operation.

Hypothesis 3: *A modern MDE workflow provides a suitable methodology for engineering hybrid simulation models in practical applications. It offers an intuitive model abstraction and formally sound way of model specification that enables modular composition, reuse and allows to derive and configure hybrid DEVS-based simulations.*

This statement can only be partly confirmed. The most notable advantages of MDE in practical application are rapid model development and increased productivity [55]. The domain-specific model abstraction allows to formalize component-based models in an intuitive manner and provides domain-specific modeling support on top of hyPDEVS. For example, domain-specific typed interfaces (for energy, entity, etc.) can be introduced that allow to check possible violations of composition rules at design time in the custom DSL editor. Unnecessary implementation details can be hidden from the user to avoid accidental complexity. For example, the `COM` channel used for acknowledging entities that is necessary in hyPDEVS, can be generated automatically from the existing information.

However, the main issue in the MDE workflow involves using simulation model component libraries. As discussed in Chapter 7, established environments, such as Eclipse EMF, do not allow for ontological instantiation, which would provide a natural way of instantiating library components within the M1 layer. While MDE is intended to reuse code and functions across different development phases and thus provide model continuity, they impose a four-layer meta-modeling architecture that requires to model the problem domain within a single meta-layer. Other approaches would require workarounds or modifying the meta-model. There is still some research required with respect to multi-level instantiation.

Although it might seem excessive to develop a formal model-driven approach for model development, it has some benefits, especially for large application projects. Most importantly, since the meta-model and corresponding DSL are tailored to the domain at hand, it allows to develop model for this domain (in our case: production systems) more efficiently by avoiding unnecessary overhead and without the need for application engineers to be M&S experts. The DSL should typically be kept lightweight and easy to understand for domain experts. This requires to balance the flexibility of the language against the efficiency of use. More flexibility inevitably entails more complexity, but allows the practitioner to model a broader range of applications.

To summarize, the methods presented in this work employ Modeling and Simulation (M&S), in particular simulation-based optimization and model-driven development, to enable the integration of energy efficiency aspects into modern PPC systems and increase energy efficiency in production.

8.2 Future Work

Regarding possible improvements, various issues may be the topic of future investigation. First and foremost, the model engineering approach presented in Chapter 7 may be extended significantly. We show only a basic proof of concept for component-based model couplings to demonstrate the general applicability. As a possible extension we had considered for example to also model the internal behavior within the hyPIM modeling layer, by means of state machines and differential equations. However, we concluded that, while this would offer significantly more flexibility for modeling different application cases, it would shift the modeling effort from the model engineer to the system engineer

(see Figure 4.3) and would require the system engineer to also be a modeling expert. This was not desirable for our intended use case. This workflow might be feasible for certain scenarios, however, it must be carefully considered on a case-by-case basis. As described above, there are also some more fundamental issues that need to be faced in order to deliver a practical component-based model engineering solution that avoids accidental complexity.

Furthermore, the optimization module (specifically its application-dependent parts) might be integrated more strongly into the hyPIM meta-model in order to avoid redundancies between simulation and optimization implementation and thus accelerate the development process. To go even further, the technology-independent model description in hyPIM also allows to derive multiple different simulation implementations from the same specification, as briefly mentioned in Section 7.3 and depicted in Figure 7.4. This includes for example simplified DE models that only focus on assessing the material flow and omit the energetic aspects with the advantage of better runtime efficiency compared to the hybrid model. A potential application might be simulation-based production planning with multiple planning stages – a long-term planning stage with long planning horizon (and simpler model for higher simulation speed) and a short-term stage with detailed simulation. While it can be feasible to employ simulation in both of these planning stages, developing and maintaining two different simulation models (of essentially the same system) is too time-consuming in practice. This is where high-level model specification with hyPIM can help, as it only requires a single unified model specification. The different implementations are then derived automatically [132].

For practical model engineering in large-scale applications, a need for sophisticated model engineering quickly arises that includes management of component variants and versions. This can also be accomplished by means of MDE [40], or, alternatively, by combining it with other methods. In this context, the System Entity Structure (SES) [243, 213, 320] has become a popular method that supports managing a model base of component models. SES is a structural knowledge representation scheme that contains decomposition, taxonomy and coupling of a system based on components.

Regarding the optimization method presented in Chapter 6, the GVNS meta-heuristic has proven to be suitable for energy-aware production planning. However, the implementation still leaves room for improvement. The performance (regarding computation speed as well as solution quality) may be further investigated and improved, for example by testing additional neighborhood operators. As an advantage, the GVNS is very flexible and easily expandable in this respect, compared to GA for example. Also, parallelization of optimization steps [264] has shown to be quite effective in boosting overall speed, which is important for practical usability.

The hybrid simulation approach based on hyPDEVS described in Chapter 5, while also proven to be promising, still requires a lot of work to develop it into an industry-grade hybrid simulation solution. Especially the need for high-level domain-specific modeling will be inevitable in practice in order to be able to manage large-scale real-life applications. However, compared to co-simulation, the advantages are evident.

The dynamic thermal behavior of the workpieces themselves has been excluded from the simulation models. Although the entities have a thermal storage capacity, their temperature changes are assumed to be instantaneous. Incorporating the dynamics of the thermal entity behavior could improve the accuracy of the simulation results. However, implementing this is not trivial and requires some effort. Some work in this regard has been conducted based on DSDEVS [78].

Possibly, an integration with the Functional Mock-up Interface (FMI) could also be considered. FMI has become quite popular for modeling physical systems and there are now a number of simulation environments that support FMI. The Modelica language and its foundation in Bond Graph theory [41] are especially suited to model energy flows. There is research being conducted [93, 94, 208] to facilitate the inclusion of discrete behavior into Modelica, in the form of (synchronous) state machines, that could perhaps one day also be included in the FMI specification. This could become of value for implementing Cubes as FMI components.

One could perhaps even imagine to realize acausal coupling relationships (similar to Modelica or MATLAB/Simscape) across Cubes. This idea as a research direction for co-simulation is also mentioned in [113]. Acausal coupling would further improve reusability of Cubes, as it would allow to instantiate them in different contexts (with different boundary conditions. However, this is still a long way to go, and there are still many challenges to be solved. In addition, our use case demonstrations have shown that causal modeling was not a major disadvantage. The material flow needs to be modeled causally anyway, and with the energetic components, especially the energy system and TBS Cubes, an acausal model would not be easy to implement. Many of our energy system Cubes models are based on empirical characteristic curves (rather than physics-based equations), which do not allow a simple change of causalization. For example, a Chiller Cube always operates in the direction of consuming electrical power and converting it into cooling energy. An operation in the opposite direction would not make sense and would not even be technically possible. However, these characteristic curve models are more runtime-efficient than acausal physics-based models, which would require additional state variables and higher level of detail to deliver a meaningful model [183]. As a consequence, our developed Cube models are always operated with the same causalization.

The Cube concept itself, which was described in Chapter 4 provides an ideal foundation to be expanded to also include other aspects besides simulation models. Of particular interest would be measurement and real-time data of the real production components, which could be managed in a component-based manner. In this way, the Cube concept might be extended to become a full-fledged digital twin of the production system [275, 288, 287, 174, 35]. Also, following the vision of Industry 4.0, Cubes could further encapsulate more intelligent control logic that takes over control tasks, thereby creating a form of autonomous agents that e.g. negotiate *Pplan* schedules among themselves instead of being specified by a higher-level (monolithic) planning module.

APPENDIX A

Oven Cube hyPDEVS Implementation

Algorithm A.1: hyPDEVS implementation of the OvenAtomic component

$ta(s) \leftarrow s.\sigma;$

$\delta_{\text{conf}}(s, e, x) \leftarrow \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x);$

function $f(s, e, x)$**:**
  // derivative of Oven temperature T
  $dT \leftarrow \left(x.\dot{Q}_h - (s.T - x.T_a) \cdot UA\right) / (c_p \cdot m_{th} + \sum_i s.ent(i).m \cdot s.ent(i).c_p);$
  return $dT;$

$c_{se}(s) \leftarrow sign \cdot (s.T - T_{set});$

function $\delta_{\text{state}}(s, e, x)$**:**
  if $s.state =$ `heating` **then**
    $s.state \leftarrow$ `waiting`;
    $s.\sigma \leftarrow 0;$
  end
  return $s;$

function $\lambda^c(s, e, x)$**:**
  $y.\dot{Q}_{wh} \leftarrow ((s.T - x.T_a) \cdot UA + x.P_{el}) \cdot (1 - \eta);$
  $y.\dot{Q}_{rec} \leftarrow ((s.T - x.T_a) \cdot UA + x.P_{el}) \cdot \eta;$
  $y.T \leftarrow s.T;$ // output current temperature for controller
  return $y;$

```
18 function δ_ext(s, e, x):
       // incoming Pplan
19     if x.Pplan and s.state ∈ {off,standby,heating,waiting} then
20         if x.Pplan < 1 then
21             s.state ← off;
22         else if x.Pplan < 2 then
23             s.state ← standby;
24         else
25             s.state ← heating;
26         end
27     end
28     s.σ ← s.σ − e;
       // incoming E_outcom, E_wcom
29     if x.E_outcom then
30         s.eoutrec ← true;
31     end
32     if x.E_wcom or α = 0 then
33         s.ewrec ← true;
34     end
35     if s.eoutrec and s.ewrec then
36         s.ent ← shift(s.ent);                    // transitory state update
37         if s.ent = ∅ then
38             s.state ← waiting;                   // switch to the next state
39             s.σ ← ∞;
40         else
41             s.state ← holding;
42             s.σ ← t_B/N;
43         end
44         s.eoutrec ← false;
45         s.ewrec ← false;
46     end
       // incoming entity on port E_in
47     if x.E_in and s.ent(1) = ∅ and s.state ∈ {waiting,holding} then
48         s.ent(1) ← x.E_in;                       // transitory state incoming
49         s.state ← holding;
50         s.σ ← 0;                                 // schedule sending E_incom
51         s.ack ← true;
52     else
53         s.σ ← s.σ − e;
54     end
55     return s;
```

```
56 function δ_int(s):
57     if s.ack then
58         s.ack ← false;                                  // finished sending E_incom
59         s.σ ← t_B/N;
60     else if s.state = holding then
61         if s.ent(N) not = ∅ then
62             s.state ← output;
63             s.σ ← 0;
64         else
65             s.ent ← shift(s.ent);                       // transitory state update
66             s.σ ← t_B/N;
67         end
68     else if s.state = output then
69         s.σ ← t_S;                                      // schedule resending entity
70     else
71         s.σ ← ∞;                                        // go passive
72     end
73     return s;

74 function λ^d(s):
75     if s.ack then
76         y.E_incom ← 1;                                  // send E_incom
77     end
78     if s.state = output then
79         if not s.eoutrec then
80             y.Eout ← s.ent(N);                          // send E_out
81             y.E_out.T ← T;
82             y.E_out.m ← E_out.m · (1 − α);
83         end
84         if not s.ewrec and α > 0 then
85             y.E_a ← s.ent(N);                           // send E_w
86             y.E_a.m ← E_a.m · α;
87         end
88     end
       // controller on/off
89     if s.state = off then
90         y.contr ← 0;
91     else if s.state = standby then
92         y.contr ← 1;
93     else
94         y.contr ← 2;
95     end
96     return y;
```

Algorithm A.2: hyPDEVS implementation of the PI controller for the Oven Cube

$ta(s) \leftarrow s.\sigma$;

$\delta_{\text{conf}}(s, e, x) \leftarrow \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x)$;

function $f(s, e, x)$**:**
    // integrator part of PI controller
    $ds \leftarrow x.T - T_{set}$;
    return ds;

$c_{se}(s) \leftarrow \emptyset$;

$\delta_{\text{state}}(s, e, x) \leftarrow \emptyset$;

function $\lambda^c(s, e, x)$**:**
    if $s.contr = 0$ **then**
        $y.P_{elD} \leftarrow 0$;
        $y.\dot{Q}_{hD} \leftarrow 0$;
    else if $s.contr = 1$ **then**
        $y.P_{elD} \leftarrow P_S$;
        $y.\dot{Q}_{hD} \leftarrow 0$;
    else
        $y.P_{elD} \leftarrow P_S$;
        // controller formula (s is the error integral (cont. state))
        $y.\dot{Q}_{hD} \leftarrow \min(P_H, K_P \cdot (T - T_{set}) + K_I \cdot s)$
    end
    return y;

function $\delta_{\text{ext}}(s, e, x)$**:**
    $s.contr \leftarrow x.contr$; // x.contr is the only discrete input
    $s.\sigma \leftarrow s.\sigma - e$;
    return s;

$\delta_{\text{int}}(s) \leftarrow \emptyset$;

$\lambda^d(s) \leftarrow \emptyset$;

Algorithm A.3: hyPDEVS implementation of the *Pplan* Source for the Oven Cube

```
1  ta(s) ← s.σ;
2  δ_conf(s, e, x) ← ∅;
3  f(s, e, x) ← ∅;
4  c_se(s) ← ∅;
5  δ_state(s, e, x) ← ∅;
6  λ^c(s, e, x) ← ∅;
7  δ_ext(s, e, x) ← ∅;
8  function δ_int(s):
9      if s.row < size(Pplan) then
10         s.σ ← Pplan(s.row).time;
11     else
12         s.σ ← ∞;
13     end
14     s.row ← s.row + 1;
15     return s;
16 λ^d(s) ← Pplan(s.row);
```

List of Figures

2.1 Energy conversion chain in industry 10
2.2 Process of PPC, complemented by S&OP. The figure also shows some of the relevant documents involved: demand plan (*Dplan*), production plan (*Pplan*) and work plan (*Aplan*). 15
2.3 Typical automation pyramid of ICT systems in an industrial facility, including APS system and location of PPC tasks 17

3.1 Classification of co-simulation couplings, adapted from [271] 27
3.2 State machine model of a thermostat controller 29
3.3 Reference model for developing simulations of production facilities. The model includes physical components (blue), information components (red) as well as planning components that provide static parameters (green). Black arrows show dynamic interactions between component variables. [130]. 33
3.4 Framework for co-simulation between MATLAB, EnergyPlus and Dymola. The software middleware BCVTB handles synchronization and data exchange [130]. 34
3.5 Comparison of annual final energy demand between three scenarios [130]. 36
3.6 Overall co-simulation implementation for the case study, including MATLAB, Dymola and EnergyPlus simulations, coupled via BCVTB middleware [130]. 38

4.1 Simulation-based planning module as part of a PPC system in the automation pyramid architecture 40
4.2 Example of a production facility consisting of different Cubes 44
4.3 Typical engineering workflow with domain engineering done by model engineers and application engineering done by system engineers, both of which are supported in the implementation by software engineers. 46
4.4 Different model engineering paradigms: specification-driven (left), formalism-driven (middle) and model-driven (right). 46
4.5 The three main modules of the overall concept: Simulation, Optimization and Model Engineering. The parts build on one another where the latter modules might be optional. 47

5.1 hyPDEVS combined model with discrete and continuous inputs, outputs and states 52

5.2 Coupled system specification using interface maps 54
5.3 Hierarchical simulator (right) mapping a hierarchical hyPDEVS model structure (left) and employing message passing between its elements 57
5.4 Generic Cube consisting of input and output interfaces for material, energy and information exchange. The internal behavior is comprised of discrete and continuous internal states and can be influenced by setting different parameters. 62
5.5 Library of Cubes models, including Production Cubes (blue), Logistics Cubes (purple), Energy Cubes (red) and Building Cubes (green) 64
5.6 Oven Cubes with interfaces and internal parameters 65
5.7 State diagram describing the discrete behavior of the Oven Cubes 66
5.8 Implementation of the Oven Cubes as hyPDEVS Coupled, consisting of three Atomics. 69
5.9 Simple example of two stations exchanging an entity, either using an acknowledgment signal (ACK) or a request signal (REQ). 73
5.10 Conceptual model of Case Study 1: Simple Production Line. It includes production machines (blue), logistics components (purple), energy supply components (red) and thermal building zones (green). 76
5.11 Numbers of entities over time in the different stations for Scenario 1a . . 79
5.12 Numbers of entities over time in the different stations for Scenario 1b . . 80
5.13 Oven (top) and Freezer (bottom) allocations for Scenario 1a (left) and Scenario 1b (right) 81
5.14 Comparison of power demand (top) and energy consumption (bottom) between Scenario 1a (left) and Scenario 1b (right) 81
5.15 Zone temperatures and ambient temperature over time for Scenario 1a . . 82
5.16 Numbers of entities over time in the different stations for Scenario 2 . . . 82
5.17 Oven (top) and Freezer (bottom) allocations for Scenario 2 83
5.18 Power demand (top) and energy consumption (bottom) for Scenario 2 . . 84
5.19 Zone temperatures and ambient temperature over time for Scenario 2 . . 84
5.20 Conceptual model of Case Study 2: Advanced Production Line with production machines (blue), logistics components (purple), energy supply components (red) and thermal building zones (green). The energy supply connections for the production machines have been omitted for reasons of clarity. 85

6.1 Simulation-optimization cycle. Starting from a demand plan (*Dplan*) of entities to be produced, the optimization algorithm generates a production schedule (*Pplan*) from the decision variables x and adapts it iteratively based on an evaluation of simulation results. 92
6.2 Exemplary production system layout with production stations, entity sources and sinks, energy supply systems and thermal zones 95
6.3 Energy price profile for Scenario 1. The price for electric energy is reduced between 10 a.m. and 4 p.m. on selected days to account for cheaper energy from a photovoltaic system during sunshine periods. 103

6.4 Progression of the cost function and part goals for Scenario 1a 104
6.5 Optimization results for Scenario 1a, initial scheduling (top) and final results (middle), both for Oven (left) and Freezer (right) allocation, as well as combined view (bottom). Dashed vertical lines indicate job due times. . . 106
6.6 Progression of the cost function and part goals for Scenario 1b 107
6.7 Oven and Freezer allocation for Scenario 1b 107
6.8 Oven and Freezer allocation for Scenario 1b without energy costs 108
6.9 Cost function and part goals for Scenario 1b without energy costs 108
6.10 Ambient temperature curve used for the simulation, based on real data . 109
6.11 Energy price profiles, constant and real 109
6.12 Optimization results for Scenario 2a, initial scheduling (top) and final results (middle), both for Oven (left) and Freezer (right) allocation, as well as combined view (bottom). The dashed vertical lines indicate job due times. 110
6.13 Cost function and part goals for Scenario 2a 111
6.14 Oven and Freezer allocation x_c for Scenario 2a without storage costs and constant energy price 112
6.15 Cost function, i.e. energy costs, for Scenario 2a without storage costs . . . 112
6.16 Oven and Freezer allocation x_r for Scenario 2a without storage costs and real energy price 113
6.17 Cost function, i.e. energy costs, for Scenario 2a without storage costs and real energy price 113
6.18 Sweeping the job starting time across the planning horizon, from 2 h (left) to 37 h (right) 114
6.19 Target value plotted over the start time sweep, both for constant energy price and real energy price profile. The red dots indicate the respective minima. 114

7.1 MDE methodology overview with two orthogonal dimensions: conceptualization (rows) and implementation (columns). Models are described using a modeling language and transformations generate executable artifacts from these models. 119
7.2 Four layers of modeling in MDE. Real-world objects are seen as instances of a model, which is an instance of a meta-mode, which is an instance of a meta-meta-model. Each model conforms to a modeling language, which is represented by a corresponding meta-level model. 122
7.3 Relationships between the different acronyms in the MDE context 124
7.4 Model development process with decreasing levels of abstraction: Conceptual Model (CM), Platform-Independent Model (PIM) and different Platform-Specific Model (PSM) implementations 126
7.5 hyPIM modeling framework with PIM, PSM as well as respective meta-models, M2M transformation and M2T code generation 127
7.6 Overview of the basic hyPIM meta-model 128
7.7 Overview of the hyPDEVS meta-model 129

7.8 Implementation of the modeling framework in Eclipse EMF. The meta-models conform to the common Ecore meta-meta-model provided by the EMF. The M2M transformation is implemented using ATL and the M2T transformations are based on Xtend. 131
7.9 An entity connection in the PIM (left) is transformed into two connections (with respective ports) to include the feedback connection (E_{incom}–E_{outcom}) [133] 132

List of Tables

3.1 Classification of methods for coupled simulation (adapted from [130]) . . 25

5.1 Defined entity attributes 62
5.2 Production schedules for Scenarios 1a and 1b 78

6.1 Neighborhood structures used in the GVNS 101
6.2 Demand plan for Scenario 1 104
6.3 Comparison between Scenario 1b with and without energy, showing final cost value using Equation (6.7) with $\omega_2 = 2$ 107
6.4 Demand plan for Scenario 2 109
6.5 Comparison of Scenario 2a between x_r and x_c, showing the cost value evaluated using the real energy price profile $c_{1,\mathrm{real}}(t)$ 112

List of Algorithms

5.1 hyPDEVS implementation of the *OvenAtomic* component 70

6.1 GVNS/SA algorithm 97
6.2 VND algorithm 101

A.1 hyPDEVS implementation of the OvenAtomic component 141
A.2 hyPDEVS implementation of the PI controller for the Oven Cube . . . 144
A.3 hyPDEVS implementation of the *Pplan* Source for the Oven Cube . . . 145

Glossary

ABM Agent-Based Modeling. 23, 26

APS Advanced Planning and Scheduling. 39, 47, 89, 117, 125, 127, 136

ASPeCT Adaptive Smoothed Production. Research project carried out at TU Wien 2017–2020. https://aspect.project.tuwien.ac.at. 74

ATL Atlas Transformation Language. 131, 132, 150

BaMa Balanced Manufacturing. Research project carried out at TU Wien 2014–2017. http://bama.ift.tuwien.ac.at. 9, 45, 74, 83, 118

BCVTB Building Controls Virtual Test Bed. Open-source software middleware for co-simulation. 34–36, 38, 147

BIM Building Information Modeling. 35

BPMN Business Process Model and Notation. 42, 129

CFP Carbon Footprint of Products. 63

CIM Computation-Independent Model. 123

Clabjects Unification of class and object concepts. Clabjects are elements with both type and instance facets. 130

closure under coupling Property of DEVS-based formalisms, ensuring that a well-defined coupling of systems in a DEVS formalism defines an atomic system in the same formalism. 56, 87, 136

CM Conceptual Model. 41, 42, 125, 126, 149

co-simulation Cooperative simulation, combines sub-models with multiple modeling methods and simulation algorithms from different simulation environments, where the sub-models exchange data during runtime via specialized communication interfaces. vi, 2–4, 6, 18, 19, 22, 24–35, 49, 50, 59, 87, 118, 126, 136, 139, 140, 155, 157

confluence A property of DE models, where all possible execution orders of simultaneous events lead to the same end result. 31

CPS Cyber-Physical Energy System. 32

CPS Cyber-Physical Production System. 157

CPS Cyber-Physical System. 28

CT Continuous-Time. 24, 25, 28–31, 33, 50

Cube Unified modularization concept for component-based development of interdisciplinary application models of industrial production facilities incorporating production machines, logistics, TBS and building hull. vi, 7, 43–46, 50, 60–69, 71–75, 83, 87, 117, 118, 125–128, 131, 132, 136, 140, 147, 148

DAE Differential-Algebraic Equation. 55

DE Discrete-Event. 22, 24–26, 28–31, 33, 36, 50, 139, 156, 157

DES Discrete-Event Simulation. 22–24, 49, 58

DESS Differential Equation System Specification. 57

DEV&DESS Discrete-Event and Differential Equation System Specification. 24, 50, 53, 55, 56, 58, 60

DEVS Discrete-Event System Specification. i, iii, 3–7, 42, 45, 46, 50–53, 55–60, 69, 73, 74, 87, 121, 128, 136, 137, 155, 156, 159, 160

DSDEVS Dynamic Structure DEVS. 74, 140

DSL Domain-Specific Language. 120, 121, 125, 126, 129, 138, 160

DSM Demand-Side Management. 13

DSML Domain-Specific Modeling Language. 120

EBNF Extended Backus-Naur Form. 122

EMF Eclipse Modeling Framework. 131–133, 138, 150, 160

energy efficiency Measures the ratio of usable energy output to energy input into a process. 11

ERP Enterprise Resource Planning. 16–18

FMI Functional Mock-up Interface. 26, 30, 31, 140

GA Genetic Algorithm. 23, 92, 116, 139

GPL General-Purpose Language. 120, 121

GVNS General Variable Neighborhood Search. 5, 96, 100, 115, 116, 139

heuristic Approximate method or procedure to solving a specific problem, typically derived from previous experiences with similar problems, that is not guaranteed to be optimal, but still feasible in practice. Where finding a global optimal solution is impossible or impractical, heuristics manage to find a satisfactory solution in a feasible time. 2, 19, 20, 24, 158

HLA High Level Architecture. 31

HTML Hypertext Markup Language. 120

HVAC Heating, Ventilation and Air Conditioning. 13, 14, 89

hybrid co-simulation Particular kind of co-simulation combining discrete and continuous modeling methods. 25, 28

hyPDEVS Hybrid PDEVS. 50, 52–61, 63, 65, 68–72, 74, 86, 87, 117, 118, 125–129, 132, 135, 136, 138, 139, 147–149, 153, 157, 160

hyPIM Framework for platform-independent modeling of hybrid simulation models. 126–128, 132, 133, 138, 139, 149

IIoT Industrial Internet of Things. 18

illegitimacy A property of DE models, where an infinite number of events occur within a finite time period, resulting in an unability to compute the model. 31

Industry 4.0 Strategic initiative to promote digital transformation in industry. The goal is to increase efficiency and productivity and achieve smart manufacturing by focusing heavily on pervasive interconnectivity and integration of technical processes (to form Cyber-Physical Production System (CPS)) for sharing and exploiting data using IIoT technologies, cloud computing and artificial intelligence. 18, 140

INFO Interdisciplinary Research for Energy Efficiency in Production (In German: Interdisziplinäre Forschung zur Energieoptimierung in Fertigungsbetrieben). Research project carried out at TU Wien 2010–2013. `http://projekt-info.org`. 32

KPI Key Performance Indicator. 12, 18, 21, 24

legitimacy Necessary and sufficient condition for a well-defined hyPDEVS system, requiring that, for any possible set of initial conditions, only a finite number of events may occur in a finite amount of time. vi, 55, 56, 160

M2M Model to Model Transformation. vii, 123, 126–129, 131, 132, 149, 150

M2T Model to Text Transformation. 123, 126, 127, 129, 131, 132, 149, 150

M&S Modeling and Simulation. 25, 42, 43, 45, 50, 125, 129, 136, 138

MATLAB Matrix Laboratory. 5, 120, 132

MBE Model-Based Engineering. 125

MBSE Model-Based Systems Engineering. 125

MDA Model-Driven Architecture. 123–125, 130

MDD Model-Driven Development. 124

MDE Model-Driven Engineering. i, iii, 3, 4, 6, 7, 117–125, 130, 132, 133, 137–139, 149

MDPE Model-Driven Product Engineering. 125

MDSE Model-Driven Software Engineering. 125

MES Manufacturing Execution System. 16–18

meta-heuristic Solution method for combinatorial optimization problems that relies on a high-level basic search strategy that enables to escape from local optima and perform a robust search to find a satisfying solution with feasible computational effort. In contrast to heuristics, the high-level search strategy is problem-independent and can be seen as an algorithmic template that defines how different low-level heuristic components interact in the search. v, vii, 2, 4–7, 19, 20, 22–24, 39, 90–92, 96, 116, 127, 135, 137, 139, 160

meta-meta-model Model that describes meta-models. Meta-models are instances of a more abstract meta-meta-model, which highlights and formalizes common properties of these meta-models. 118, 121–123, 129, 131, 149, 150

meta-model Model that describes models. Models are instances of a more abstract meta-model, which highlights and formalizes common properties of these models. i, 5, 6, 118, 121–123, 126–133, 138, 139, 149, 150, 158

MIC Model-Integrated Computing. 123

middleware Software system that links different simulation unit to form a co-simulation. The middleware handles orchestration and data exchange between the simulation units at runtime. 24, 30, 34, 35, 38, 118, 126, 147

MIP Mixed-Integer Linear Programming. 21

MIP Mixed-Integer Programming. 19, 90

MoC Model of Computation. 30

MOF Meta-Object Facility. 123

multi-formalism modeling . 27, *see* multi-method modeling

multi-method modeling Combines sub-models using different modeling approaches or methods to form a combined overall simulation model. 27

OCL Object Constraint Language. 123, 131

ODE Ordinary Differential Equation. 22, 25, 27, 50, 52, 57–60, 74

OMG Object Management Group. 123, 124

OOP Object-Oriented Programming. 43, 45, 118

OPC Open Platform Communications, originally OLE (Object Linking and Embedding) for Process Control. 18, 159

OPC UA OPC Unified Architecture. 18

PDA Production Data Acquisition. 17

PDEVS Parallel DEVS. 50, 52, 53, 57–59, 157

Peelboard Steel baking trays used in industrial-style bakeries for proofing, baking and deep-freezing. 85

PIM Platform-Independent Model. 123, 125–132, 149, 150

PLC Programmable Logic Controller. 17

PPC Production Planning and Control. i, vi, 1, 3, 7, 9–11, 14, 16, 18, 22, 39, 40, 42, 44, 46, 47, 89, 105, 117, 125, 136–138, 147, 159

PPS Produktionsplanung und -steuerung, see PPC. iii

PSM Platform-Specific Model. 123, 125–129, 131, 132, 149

QSS Quantized-State Systems. 29, 58, 60, 75

S&OP Sales and Operations Planning. 14

SA Simulated Annealing. 5, 7, 19, 20, 97, 102, 105, 109

SCADA Supervisory Control and Data Acquisition. 17, 21

SD System Dynamics. 23, 26

SEC Specific Energy Consumption. Ratio between energy input to physical output of a process. 12

semantic adaptation Adaptation of time, control and data between different heterogeneous modeling formalisms. 28, 30

SES System Entity Structure. 139

simheuristic Particular simulation-based optimization approach employing meta-heuristic methods, oriented to efficiently tackle a combinatorial optimization problem instance. Often used in combination with stochastic components [151]. 92

smart grid Intelligent electrical power grid that is integrated with a bidirectional communication network to allow real-time monitoring and distributed control. 32

SQL Structured Query Language. 120

state event legitimacy Extension of the legitimacy property for hybrid DEVS. 56

SysML Systems Modeling Language. 42, 45, 69

TBS Technical Building Services. 13, 14, 22, 23, 34, 36, 75, 80, 83, 89, 90, 113, 140, 156

transitory state A state s in a hyPDEVS atomic for which $ta(s) = 0$, meaning that, upon entry, the state is left immediatel. 56, 65

UML Unified Modeling Language. 42, 120, 121

VHDL Very High Speed Integrated Circuit Hardware Description Language. 120

VND Variable Neighborhood Descent. 5, 96, 97, 101

VNS Variable Neighborhood Search. i, iii, 5, 7, 19, 21, 91, 96, 97, 100–102, 104, 115, 116

Xtend High-level programming language that can be used for code generation within the Eclipse EMF framework. 131, 132, 150

Xtext Framework for developing programming languages and DSLs. Part of the Eclipse EMF framework. 131

Bibliography

[1] Foued Ben Abdelaziz, Saoussen Krichen, and Jouhaina Chaouachi. "A Hybrid Heuristic for Multiobjective Knapsack Problems". In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Ed. by Stefan Voß, Silvano Martello, Ibrahim H. Osman, and Catherine Roucairol. Boston, MA: Springer US, 1999, pp. 205–212. ISBN: 978-1-4615-5775-3. DOI: 10.1007/978-1-4615-5775-3_14. URL: https://doi.org/10.1007/978-1-4615-5775-3_14 (visited on Mar. 18, 2020).

[2] Nabil Absi and Safia Kedad-Sidhoum. "MIP-Based Heuristics for Multi-Item Capacitated Lot-Sizing Problem with Setup Times and Shortage Costs". In: *RAIRO - Operations Research* 41.2 (Apr. 2007), pp. 171–192. ISSN: 0399-0559, 1290-3868. DOI: 10.1051/ro:2007014. URL: https://www.cambridge.org/core/journals/rairo-operations-research/article/mipbased-heuristics-for-multiitem-capacitated-lotsizing-problem-with-setup-times-and-shortage-costs/5E9FC35CA4A27C940234723E5886BC09 (visited on Mar. 18, 2020).

[3] M. A. Adibi, M. Zandieh, and M. Amiri. "Multi-Objective Scheduling of Dynamic Job Shop Using Variable Neighborhood Search". In: *Expert Systems with Applications* 37.1 (Jan. 1, 2010), pp. 282–287. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.05.001. URL: http://www.sciencedirect.com/science/article/pii/S0957417409004199 (visited on Mar. 8, 2020).

[4] Carlos Agostinho, Filipe Correia, and Ricardo Jardim-Goncalves. "Interoperability of Complex Business Networks by Language Independent Information Models". In: *New World Situation: New Directions in Concurrent Engineering*. Ed. by Jerzy Pokojski, Shuichi Fukuda, and Józef Salwiński. Advanced Concurrent Engineering. London: Springer, 2010, pp. 111–124. ISBN: 978-0-85729-024-3. DOI: 10.1007/978-0-85729-024-3_12.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. -H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. "The Algorithmic Analysis of Hybrid Systems". In: *Theoretical Computer Science*. Hybrid Systems 138.1 (Feb. 6, 1995), pp. 3–34. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)00202-T. URL: http://www.sciencedirect.com/science/article/pii/030439759400202T (visited on Feb. 11, 2020).

[6] Leonardus Budiman Arief. “A Framework for Supporting Automatic Simulation Generation from Design”. Thesis. Newcastle University, 2001. URL: http://theses.ncl.ac.uk/jspui/handle/10443/1816 (visited on May 4, 2020).

[7] C. Atkinson. “Meta-Modelling for Distributed Object Environments”. In: *Proceedings First International Enterprise Distributed Object Computing Workshop*. Proceedings First International Enterprise Distributed Object Computing Workshop. Oct. 1997, pp. 90–101. DOI: 10.1109/EDOC.1997.628350.

[8] Colin Atkinson and Thomas Kühne. “Demystifying Ontological Classification in Language Engineering”. In: *Modelling Foundations and Applications*. Ed. by Andrzej Wąsowski and Henrik Lönn. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 83–100. ISBN: 978-3-319-42061-5. DOI: 10.1007/978-3-319-42061-5_6.

[9] Colin Atkinson and Thomas Kühne. “Model-Driven Development: A Metamodeling Foundation”. In: *IEEE Software* 20.5 (Sept. 2003), pp. 36–41. ISSN: 1937-4194. DOI: 10.1109/MS.2003.1231149.

[10] Colin Atkinson and Thomas Kühne. “Processes and Products in a Multi-Level Metamodeling Architecture”. In: *International Journal of Software Engineering and Knowledge Engineering* 11 (2001), pp. 761–783.

[11] Colin Atkinson and Thomas Kühne. “Reducing Accidental Complexity in Domain Models”. In: *Software & Systems Modeling* 7.3 (July 1, 2008), pp. 345–359. ISSN: 1619-1374. DOI: 10.1007/s10270-007-0061-0. URL: https://doi.org/10.1007/s10270-007-0061-0 (visited on May 11, 2020).

[12] Colin Atkinson and Thomas Kühne. “The Essence of Multilevel Metamodeling”. In: *International Conference on the Unified Modeling Language*. International Conference on the Unified Modeling Language. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Oct. 1, 2001, pp. 19–33. ISBN: 978-3-540-42667-7. DOI: 10.1007/3-540-45441-1_3. URL: https://link.springer.com/chapter/10.1007/3-540-45441-1_3 (visited on Feb. 16, 2018).

[13] Colin Atkinson and Thomas Kühne. “The Role of Metamodeling in MDA”. In: *Proceedings of the UML 2002 Workshop on Software Model Engineering*. 2002, pp. 67–70.

[14] Colin Atkinson, Thomas Kühne, and Juan de Lara. “Editorial to the Theme Issue on Multi-Level Modeling”. In: *Software & Systems Modeling* 17.1 (Feb. 1, 2018), pp. 163–165. ISSN: 1619-1374. DOI: 10.1007/s10270-016-0565-6. URL: https://doi.org/10.1007/s10270-016-0565-6 (visited on May 11, 2020).

[15] Colin Atkinson, Thomas Kühne, and Brian Henderson-Sellers. “To Meta or Not to Meta - That Is the Question”. In: *Journal of Object-Oriented Programming* 13.8 (2000), pp. 32–35. URL: https://www.academia.edu/15638189/To_Meta_or_Not_to_Meta._That_Is_the_Question (visited on Nov. 19, 2019).

[16] Muhammad Usman Awais. “Distributed Hybrid Co-Simulation”. Dissertation. Wien: TU Wien, 2015. URL: http://media.obvsg.at/p-AC12706426-2001 (visited on Oct. 24, 2016).

[17] Muhammad Usman Awais, Peter Palensky, Wolfgang Mueller, Edmund Widl, and Atiyah Elsheikh. “Distributed Hybrid Simulation Using the HLA and the Functional Mock-up Interface”. In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society. Vienna, Austria: IEEE, Nov. 2013, pp. 7564–7569. ISBN: 978-1-4799-0224-8. DOI: 10.1109/IECON.2013.6700393. URL: http://ieeexplore.ieee.org/document/6700393/ (visited on Feb. 12, 2020).

[18] A. M. Bagirov, B. Karasözen, and M. Sezer. “Discrete Gradient Method: Derivative-Free Method for Nonsmooth Optimization”. In: *Journal of Optimization Theory and Applications* 137.2 (May 1, 2008), pp. 317–334. ISSN: 1573-2878. DOI: 10.1007/s10957-007-9335-5. URL: https://doi.org/10.1007/s10957-007-9335-5 (visited on Mar. 18, 2020).

[19] Sergey Balandin, Michel Gillet, Irina Lavrovskaya, Valentin Olenev, Alexey Rabin, and Alexander Stepanov. “Co-Modeling of Embedded Networks Using SystemC and SDL”. In: *International Journal of Embedded and Real-Time Communication Systems* 2.1 (Jan. 2011), pp. 23–48. ISSN: 1947-3176, 1947-3184. DOI: 10.4018/jertcs.2011010102. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jertcs.2011010102 (visited on Feb. 5, 2020).

[20] O Balci and W F Ormsby. “Conceptual Modelling for Designing Large-Scale Simulations”. In: *Journal of Simulation* 1.3 (Aug. 1, 2007), pp. 175–186. ISSN: 1747-7778. DOI: 10.1057/palgrave.jos.4250023. URL: https://orsociety.tandfonline.com/doi/full/10.1057/palgrave.jos.4250023 (visited on May 4, 2020).

[21] O. Balci, J. D. Arthur, and R. E. Nance. “Accomplishing Reuse with a Simulation Conceptual Model”. In: *2008 Winter Simulation Conference*. 2008 Winter Simulation Conference. Dec. 2008, pp. 959–965. DOI: 10.1109/WSC.2008.4736162.

[22] Fernando 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. URL: http://dl.acm.org/citation.cfm?id=224731 (visited on Oct. 30, 2016).

[23] Richard E. Bellman and Stuart E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Dec. 8, 2015. 389 pp. ISBN: 978-1-4008-7465-1. Google Books: ZgbWCgAAQBAJ.

[24] F. Bergero and E. Kofman. “PowerDEVS: A Tool for Hybrid System Modeling and Real-Time Simulation”. In: *SIMULATION* 87.1-2 (Jan. 1, 2011), pp. 113–132. ISSN: 0037-5497, 1741-3133. DOI: 10.1177/0037549710368029. URL: http://sim.sagepub.com/cgi/doi/10.1177/0037549710368029 (visited on Oct. 31, 2016).

[25] Lorenzo Bettini. *Implementing Domain-Specific Languages with Xtext and Xtend*. Community Experience Distilled. Birmingham: Packt Publishing, 2013. 324 pp. ISBN: 978-1-78216-030-4. URL: https://dl.acm.org/citation.cfm?id=3074444.

[26] Michael Bierbrauer, Christian Menn, Svetlozar T. Rachev, and Stefan Trück. “Spot and Derivative Pricing in the EEX Power Market”. In: *Journal of Banking & Finance*. Risk Management and Quantitative Approaches in Finance 31.11 (Nov. 1, 2007), pp. 3462–3485. ISSN: 0378-4266. DOI: 10.1016/j.jbankfin.2007.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0378426607001355 (visited on Mar. 31, 2020).

[27] Louis G. Birta and Gilbert Arbez. *Modelling and Simulation: Exploring Dynamic System Behaviour*. 2nd ed. Simulation Foundations, Methods and Applications. London: Springer-Verlag, 2013. ISBN: 978-1-4471-6836-2. DOI: 10.1007/978-1-4471-2783-3. URL: https://www.springer.com/de/book/9781447168362 (visited on May 4, 2020).

[28] Friedrich Bleicher, Fabian Dür, Ines Leobner, Iva Kovacic, Bernhard Heinzl, and Wolfgang Kastner. “Co-Simulation Environment for Optimizing Energy Efficiency in Production Systems”. In: *CIRP Annals – Manufacturing Technology* 63.1 (Aug. 2014), pp. 441–444. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2014.03.122. URL: http://www.sciencedirect.com/science/article/pii/S0007850614001255 (visited on Sept. 26, 2016).

[29] Friedrich Bleicher, Benjamin Mörzinger, Christoph Loschan, Ines Leobner, Peter Smolek, Wolfgang Kastner, Bernhard Heinzl, Iva Kovacic, Georgios Gourlis, Thomas Sobottka, Felix Kamhuber, Roman Klug, AutomationX GmbH, Philipp Trummer, AutomationX GmbH, Michael Mühlhauser, AutomationX GmbH, Neat Likaj, AutomationX GmbH, Gerhard Gotz, Klemens Gregor Schulmeister, Günther Daubner, Rudolf Zeman, Niki Popper, Matthias Rössler, Thomas Palatin, Erich Krall, Josef Obiltschnig, Karsten Buchholz, Ewald Perwög, Friedrich Koidl, Gerhard Fritz, and Albert Dechant. *Balanced Manufacturing (BaMa) – Final Report*. Vienna, Austria: TU Wien, 2018, p. 73. URL: http://bama.ift.tuwien.ac.at/static/downloadfiles/02_Bama_publizierbarer_endbericht.pdf (visited on July 4, 2020).

[30] Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. “Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models”. In:

Proceedings of the 9th International Modelica Conference. Munich, Germany, 2012, pp. 173–184. DOI: 10.3384/ecp12076173. URL: http://www.ep.liu.se/ecp_article/index.en.aspx?issue=76;article=17 (visited on May 11, 2017).

[31] Christian Blum, Jakob Puchinger, Günther R. Raidl, and Andrea Roli. "Hybrid Metaheuristics in Combinatorial Optimization: A Survey". In: *Applied Soft Computing* 11.6 (Sept. 1, 2011), pp. 4135–4151. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2011.02.032. URL: http://www.sciencedirect.com/science/article/pii/S1568494611000962 (visited on Mar. 18, 2020).

[32] Jean-Sébastien Bolduc and Hans Vangheluwe. "Mapping {ODEss to {DEVS Adaptive Quantization". In: *Proceedings of the 2003 Summer Computer Simulation Conference (SCSC 2003)*. Montreal, Quebec, Canada: SCS, 2003, pp. 401–407. ISBN: 1-56555-268-7.

[33] Eric Bonneville and Anne Rialhe. "Good Practice for Energy Efficiency in Industry". In: *Efficiency & Ecodesign* (2006). URL: http://www.leonardo-energy.info/sites/leonardo-energy/files/root/Documents/2009/DSM-industry.pdf (visited on Nov. 10, 2016).

[34] Martin Bornschlegl. *Methods-Energy Measurement: eine Methode zur Energieplanung für Fügeverfahren im Karosseriebau*. Fertigungstechnik - Erlangen 285. Bamberg: Meisenbach GmbH Verlag, 2016. 136 pp. ISBN: 978-3-87525-409-9.

[35] Stefan Boschert and Roland Rosen. "Digital Twin—The Simulation Aspect". In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*. Ed. by Peter Hehenberger and David Bradley. Cham: Springer International Publishing, 2016, pp. 59–74. ISBN: 978-3-319-32156-1. DOI: 10.1007/978-3-319-32156-1_5. URL: https://doi.org/10.1007/978-3-319-32156-1_5 (visited on June 8, 2020).

[36] 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. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4062043 (visited on Oct. 28, 2016).

[37] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. "A Survey on Optimization Metaheuristics". In: *Information Sciences*. Prediction, Control and Diagnosis Using Advanced Neural Computations 237 (July 10, 2013), pp. 82–117. ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.02.041. URL: http://www.sciencedirect.com/science/article/pii/S0020025513001588 (visited on Mar. 8, 2020).

[38] Uwe Bracht, Dieter Geckler, and Sigrid Wenzel. *Digitale Fabrik: Methoden und Praxisbeispiele*. 2., aktualisierte und erweiterte Auflage. VDI-Buch. Berlin: Springer Vieweg, 2018. 475 pp. ISBN: 978-3-662-55783-9.

[39] Sally C. Brailsford, Tillal Eldabi, Martin Kunc, Navonil Mustafee, and Andres F. Osorio. "Hybrid Simulation Modelling in Operational Research: A State-of-the-Art Review". In: *European Journal of Operational Research* 278.3 (Nov. 1, 2019), pp. 721–737. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.10.025. URL: http://www.sciencedirect.com/science/article/pii/S0377221718308786 (visited on Jan. 29, 2020).

[40] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice*. 1st ed. Vol. 1. Synthesis Lectures on Software Engineering. Morgan & Claypool, Sept. 21, 2012. ISBN: 978-1-60845-882-0. URL: https://doi.org/10.2200/S00441ED1V01Y201208SWE001 (visited on May 23, 2018).

[41] Jan F. Broenink. "Object–Oriented Modeling with Bond Graphs and Modelica". In: *Proceedings of the '99 International Conference on Bond Graph Modeling and Simulation, Part of the WMC '99 Western MultiConference*. San Francisco, CA, USA, Jan. 17–20, 1999, p. 6. URL: https://modelica.org/publications/papers/icbgm99bnkw.pdf.

[42] Jan F. Broenink, Peter-Jan D. Vos, Zhou Lu, and Maarten M. Bezemer. "A Co-Design Approach for Embedded Control Software of Cyber-Physical Systems". In: *2016 11th System of Systems Engineering Conference (SoSE)*. 2016 11th System of Systems Engineering Conference (SoSE). June 2016, pp. 1–5. DOI: 10.1109/SYSOSE.2016.7542927.

[43] David Broman, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. "Requirements for Hybrid Cosimulation Standards". In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC '15. Seattle, Washington: ACM Press, Apr. 14, 2015, pp. 179–188. ISBN: 978-1-4503-3433-4. DOI: 10.1145/2728606.2728629. URL: https://doi.org/10.1145/2728606.2728629 (visited on Feb. 6, 2020).

[44] Alan W. Brown. "An Overview of Components and Component-Based Development". In: *Advances in Computers*. Vol. 54. Elsevier, 2002, pp. 1–34. ISBN: 978-0-12-012154-0. DOI: 10.1016/S0065-2458(01)80014-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0065245801800140 (visited on May 11, 2020).

[45] Alan W. Brown. "An Overview of Components and Component-Based Development". In: *Advances in Computers*. Ed. by Marvin V. Zelkowitz. Vol. 54. Trends in Software Engineering. Elsevier, Jan. 1, 2002, pp. 1–34. DOI: 10.1016/S0065-2458(01)80014-0. URL: http://www.sciencedirect.com/science/article/pii/S0065245801800140 (visited on May 13, 2020).

[46] Bernd Bruegge and Allen A. Dutoit. *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. USA: Prentice Hall PTR, 1999. 553 pp. ISBN: 978-0-13-489725-7.

[47] Katharina Bunse, Matthias Vodicka, Paul Schönsleben, Marc Brülhart, and Frank O. Ernst. "Integrating Energy Efficiency Performance in Production Management – Gap Analysis between Industrial Needs and Scientific Literature". In: *Journal of Cleaner Production* 19.6–7 (Apr. 2011), pp. 667–679. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2010.11.011. URL: http://www.sciencedirect.com/science/article/pii/S0959652610004452 (visited on Oct. 3, 2016).

[48] Ansgar Burger, Andreas Lang, and Yannis Müller. "Mögliche Veränderungen Von System-Architekturen Im Bereich Der Produktion". In: *Industrie 4.0: Wie Cyber-Physische Systeme Die Arbeitswelt Verändern*. Ed. by Volker P. Andelfinger and Till Hänisch. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, pp. 57–68. ISBN: 978-3-658-15557-5. DOI: 10.1007/978-3-658-15557-5âĆĎ. URL: https://doi.org/10.1007/978-3-658-15557-5%E2%82%84.

[49] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. "A Classification of Hyper-Heuristic Approaches". In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2010, pp. 449–468. ISBN: 978-1-4419-1665-5. DOI: 10.1007/978-1-4419-1665-5_15. URL: https://doi.org/10.1007/978-1-4419-1665-5_15 (visited on Mar. 28, 2020).

[50] Martin Busch, Martin Arnold, Andreas Heckmann, and S. Dronka. "Interfacing SIMPACK to Modelica/Dymola for Multi-Domain Vehicle System Simulations". In: *SIMPACK News* 11.2 (2007), pp. 1–3. URL: http://simpack.com/fileadmin/simpack/doc/newsletter/2007/SN-2-2007_n.pdf (visited on Oct. 29, 2016).

[51] Martin Büscher, Arno Claassen, Matthias Kube, Sebastian Lehnhoff, Klaus Piech, Sebastian Rohjans, Stefan Scherfke, Cornelius Steinbrink, Jorge Velasquez, Francois Tempez, et al. "Integrated Smart Grid Simulations for Generic Automation Architectures with RT-LAB and Mosaik". In: *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference On*. IEEE, 2014, pp. 194–199. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7007645 (visited on Oct. 6, 2016).

[52] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. *Languages and Tools for Hybrid Systems Design*. Vol. 1. Foundations and Trends in Electronic Design Automation 1-2. Boston, 2006. 194 pp. URL: http://dx.doi.org/10.1561/1000000001 (visited on Feb. 11, 2020).

[53] Yolanda Carson and Anu Maria. "Simulation Optimization: Methods and Applications". In: *Proceedings of the 29th Winter Simulation Conference*. WSC '97. Atlanta, Georgia, USA: IEEE Computer Society, Dec. 1, 1997, pp. 118–126. ISBN: 978-0-7803-4278-1. DOI: 10.1145/268437.268460. URL: https://doi.org/10.1145/268437.268460 (visited on Mar. 29, 2020).

[54] A. Cataldo, M. Taisch, and B. Stahl. "Modelling, Simulation and Evaluation of Energy Consumptions for a Manufacturing Production Line". In: *Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON)*. IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society. Vienna, Austria, Nov. 2013, pp. 7537–7542. DOI: 10.1109/IECON.2013.6700388.

[55] Deniz Cetinkaya. "Model Driven Development of Simulation Models: Defining and Transforming Conceptual Models into Simulation Models by Using Metamodels and Model Transformations". Delft University of Technology, 2013. DOI: 10.4233/UUID:3DB45913-1662-429F-A385-ED53F5AC41FD. URL: http://resolver.tudelft.nl/uuid:3db45913-1662-429f-a385-ed53f5ac41fd (visited on May 3, 2020).

[56] Deniz Cetinkaya and Alexander Verbraeck. "Metamodeling and Model Transformations in Modeling and Simulation". In: *Proceedings of the 2011 Winter Simulation Conference (WSC)*. Proceedings of the 2011 Winter Simulation Conference (WSC). Dec. 2011, pp. 3043–3053. DOI: 10.1109/WSC.2011.6148005.

[57] Deniz Cetinkaya, Alexander Verbraeck, and Mamadou D Seck. "Model Transformation from BPMN to DEVS in the MDD4MS Framework". In: *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium*. Orlando, FL, USA: Society for Computer Simulation International, Mar. 26–29, 2012, p. 6. URL: https://dl.acm.org/citation.cfm?id=2346644.

[58] Deniz Cetinkaya, Alexander Verbraeck, and Mamadou D. Seck. "Applying a Model Driven Approach to Component Based Modeling and Simulation". In: *Proceedings of the Winter Simulation Conference*. WSC '10. Baltimore, Maryland, 2010, pp. 546–553. ISBN: 978-1-4244-9864-2. URL: http://dl.acm.org/citation.cfm?id=2433508.2433571 (visited on July 1, 2018).

[59] Yeen Chan, Ravi Kantamaneni, and Mark Allington. *Study on Energy Efficiency and Energy Saving Potential in Industry and on Possible Policy Mechanisms*. ICF Cons. Ltd., Dec. 2015. URL: https://ec.europa.eu/energy/en/studies/study-energy-efficiency-and-energy-saving-potential-industry-and-possible-policy-mechanisms (visited on Jan. 29, 2018).

[60] Peter Checkland. *Soft Systems Methodology: A 30-Year Retrospective*. Chichester, New York: John Wiley, 1999. 1 p. ISBN: 978-0-471-98606-5.

[61] Guorong Chen, Liang Zhang, Jorge Arinez, and Stephan Biller. "Energy-Efficient Production Systems Through Schedule-Based Operations". In: *IEEE Transactions on Automation Science and Engineering* 10.1 (Jan. 2013), pp. 27–37. ISSN: 1558-3783. DOI: 10.1109/TASE.2012.2202226.

[62] Xi Chen, Hongxing Yang, and Ke Sun. "Developing a Meta-Model for Sensitivity Analyses and Prediction of Building Performance for Passively Designed High-Rise Residential Buildings". In: *Applied Energy* 194 (May 15, 2017), pp. 422–439. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2016.08.180. URL: http://www.sciencedirect.com/science/article/pii/S0306261916312892 (visited on May 3, 2020).

[63] Manuel Chica, Angel A. Juan Pérez, Oscar Cordon, and David Kelton. *Why Simheuristics? Benefits, Limitations, and Best Practices When Combining Metaheuristics with Simulation*. SSRN Scholarly Paper ID 2919208. Rochester, NY: Social Science Research Network, Jan. 1, 2017. DOI: 10.2139/ssrn.2919208. URL: http://dx.doi.org/10.2139/ssrn.2919208 (visited on Mar. 31, 2020).

[64] Alex Chung Hen Chow and Bernard P. Zeigler. "Parallel DEVS: A Parallel, Hierarchical, Modular, Modeling Formalism". In: *Proceedings of the 1994 Winter Simulation Conference*. Winter Simulation Conference. WSC '94. San Diego, CA, USA: Society for Computer Simulation International, 1994, pp. 716–722. ISBN: 978-0-7803-2109-0. URL: http://dl.acm.org/citation.cfm?id=193201.194336 (visited on Oct. 30, 2016).

[65] Rick Chow. "Evolving Genotype to Phenotype Mappings with a Multiple-Chromosome Genetic Algorithm". In: *Genetic and Evolutionary Computation – GECCO 2004*. Ed. by Kalyanmoy Deb. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 1006–1017. ISBN: 978-3-540-24854-5. DOI: 10.1007/978-3-540-24854-5_100.

[66] Selim Ciraci, Jeff Daily, and Jason Fuller. "FNCS: A Framework for Power System and Communication Networks Co-Simulation". In: *Proceedings Symposium on Theory of Modeling & Simulation-DEVS Integrative* 46.4 (2014), pp. 256–263. ISSN: 07359276.

[67] Fabio Cremona, Marten Lohstroh, Stavros Tripakis, Christopher Brooks, and Edward A. Lee. "FIDE: An FMI Integrated Development Environment". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. Pisa, Italy: Association for Computing Machinery, Apr. 4, 2016, pp. 1759–1766. ISBN: 978-1-4503-3739-7. DOI: 10.1145/2851613.2851677. URL: https://doi.org/10.1145/2851613.2851677 (visited on Feb. 10, 2020).

[68] I. Crnkovic, M. Chaudron, and S. Larsson. "Component-Based Development Process and Component Lifecycle". In: *International Conference on Software Engineering Advances*. International Conference on Software Engineering Advances. Oct. 2006. DOI: 10.1109/ICSEA.2006.261300.

[69] Krzysztof Czarnecki and Simon Helsen. "Classification of Model Transformation Approaches". In: *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*. Vol. 45. Anaheim,

California, USA, 2003, p. 17. URL: https://pdfs.semanticscholar.org/1708/ac2f77e16cea1b36ac519cdc40ecd19c7cd4.pdf.

[70] Maria da Graça Carvalho. "EU Energy and Climate Change Strategy". In: *Energy* 40.1 (Apr. 1, 2012), pp. 19–22. ISSN: 0360-5442. DOI: 10.1016/j.energy.2012.01.012. URL: http://www.sciencedirect.com/science/article/pii/S0360544212000175 (visited on Feb. 21, 2020).

[71] Judith S. Dahmann, Richard M. Fujimoto, and Richard M. Weatherly. "The Department of Defense High Level Architecture". In: *Proceedings of the 1997 Winter Simulation Conference*. IEEE Computer Society, 1997, pp. 142–149. URL: http://dl.acm.org/citation.cfm?id=268465 (visited on Apr. 28, 2017).

[72] Dassault Systèmes AB. *Dymola Dynamic Modeling Laboratory: Getting Started with Dymola*. Version 2020. Mar. 2019. URL: http://www.3ds.com/fileadmin/PRODUCTS/CATIA/DYMOLA/PDF/Getting_started_with_Dymola.pdf (visited on Feb. 18, 2020).

[73] Tatjana Davidovic and Teodor Gabriel Crainic. *Parallelization strategies for variable neighborhood search*. CIRRELT-2013-47. CIRRELT – Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, 2013. URL: https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2013-47.pdf (visited on Mar. 28, 2020).

[74] Tatjana Davidović and Teodor Gabriel Crainic. "MPI Parallelization of Variable Neighborhood Search". In: *Electronic Notes in Discrete Mathematics*. EURO Mini Conference 39 (Dec. 1, 2012), pp. 241–248. ISSN: 1571-0653. DOI: 10.1016/j.endm.2012.10.032. URL: http://www.sciencedirect.com/science/article/pii/S1571065312000339 (visited on Mar. 28, 2020).

[75] Jim Davis, Thomas Edgar, James Porter, John Bernaden, and Michael Sarli. "Smart Manufacturing, Manufacturing Intelligence and Demand-Dynamic Performance". In: *Computers & Chemical Engineering*. FOCAPO 2012 47 (Dec. 20, 2012), pp. 145–156. ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2012.06.037. URL: http://www.sciencedirect.com/science/article/pii/S0098135412002219 (visited on Mar. 1, 2020).

[76] Christina Deatcu, Birger Freymann, and Thorsten Pawletta. "PDEVS-Based Hybrid System Simulation Toolbox for MATLAB". In: *Proceedings of the Symposium on Theory of Modeling & Simulation (SpringSim-TMS/DEVS 2017)*. Virginia Beach, VA, USA: Society for Computer Simulation International (SCS), Apr. 23–26, 2017. URL: http://scs.org/wp-content/uploads/2017/06/13_Final_Manuscript-4.pdf (visited on Jan. 17, 2018).

[77] Christina Deatcu and Thorsten Pawletta. "A Qualitative Comparison of Two Hybrid DEVS Approaches". In: *SNE - Simulation Notes Europe* 22.1 (2012), pp. 15–24. DOI: 10.11128/sne.22.tn.10107. URL: http://www.sne-journal.org/fileadmin/user_upload/tx_pubdb/10107.sne.22.tn_1.pdf.

[78] Christina Deatcu, Thorsten Pawletta, Olaf Hagendorf, and Bernhard Lampe. "Considering Workpieces as Integral Parts of a DEVS Model". In: *Proceedings of the 21st European Modeling & Simulation Symposium, EMSS 2009*. Ed. by Rosa M. Aguilar and Universidad de La Laguna. Puerto de La Cruz, Spain, Sept. 23–25, 2009, pp. 27–35. ISBN: 978-84-692-5414-1. URL: http://www.msc-les.org/proceedings/content/emss2009/Considering%20Workpieces%20As%20Integral%20Parts%20Of%20A%20Devs%20Model.

[79] Bose Debayan. "Component Based Development-Application in Software Engineering". In: *Indian Statistical Institute* (2011).

[80] Yamille del Valle, Ganesh Kumar Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez, and Ronald G. Harley. "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems". In: *IEEE Transactions on Evolutionary Computation* 12.2 (Apr. 2008), pp. 171–195. ISSN: 1941-0026. DOI: 10.1109/TEVC.2007.896686.

[81] Joachim Denil, Bart Meyers, Paul De Meulenaere, and Hans Vangheluwe. "Explicit Semantic Adaptation of Hybrid Formalisms for FMI Co-Simulation". In: *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (TMS-DEVS)*. Alexandria, VA, USA, 2015, pp. 99–106. URL: https://pdfs.semanticscholar.org/ecc6/8821070b60c3d505a1ec7e6875e316aec596.pdf.

[82] Berend Denkena and Hans Kurt Tönshoff. *Spanen: Grundlagen*. 3., bearb. und erw. Aufl. VDI-Buch. Berlin: Springer, 2011. 426 pp. ISBN: 978-3-642-19771-0.

[83] Tom Devoldere, Wim Dewulf, Wim Deprez, and Joost R. Duflou. "Energy Related Life Cycle Impact and Cost Reduction Opportunities in Machine Design: The Laser Cutting Case". In: *Proceedings of the 15th CIRP Conference on Life Cycle Engineering*. Sydney, Australia: CIRP, 2008, pp. 412–419. ISBN: 1-877040-67-3.

[84] Tom Devoldere, Wim Dewulf, Wim Deprez, Barbara Willems, and Joost R. Duflou. "Improvement Potential for Energy Consumption in Discrete Part Production Machines". In: *Proceedings of the 14th CIRP Conference on Life Cycle Engineering*. Ed. by Shozo Takata and Yasushi Umeda. Tokyo, Japan: CIRP, 2007, pp. 311–316. ISBN: 978-1-84628-935-4. DOI: 10.1007/978-1-84628-935-4_54.

[85] P H Divshali, M Laukkanen, R Bhandia, TU Delft, R Bhandia, E Widl, C Steinbrink, A Kulmala, and K Mäki. "Smart Grid Co-Simulation by Developing an Fmi-Compliant Interface for Pscad". In: *Proceedings of the 25th International Conference on Electricity Distribution (CIRED 2019)*. Madrid: AIM, June 3–6, 2019, p. 6.

[86] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. "Ant Colony Optimization". In: *IEEE Computational Intelligence Magazine* 1.4 (Nov. 2006), pp. 28–39. ISSN: 1556-6048. DOI: 10.1109/MCI.2006.329691.

[87] Joost R. Duflou, John W. Sutherland, David Dornfeld, Christoph Herrmann, Jack Jeswiet, Sami Kara, Michael Hauschild, and Karel Kellens. "Towards Energy and Resource Efficient Manufacturing: A Processes and Systems Approach". In: *CIRP Annals-Manufacturing Technology* 61.2 (2012), pp. 587–609. URL: http://www.sciencedirect.com/science/article/pii/S0007850612002016 (visited on Oct. 31, 2016).

[88] Chuck Eastman, Kathleen Liston, Rafael Sacks, and Kathleen Liston. *BIM Handbook*. Ed. by Charles M. Eastman. Hoboken, N.J: Wiley, 2008. 20–21; 65–84; 93–135. ISBN: 978-0-470-18528-5. URL: http://s3.amazonaws.com/academia.edu.documents/31207284/BIM_Handbook_1st.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1476630625&Signature=bPQf4Rpas5g05QebDfp%2FJ3UKfhw%3D&response-content-disposition=inline%3B%20filename%3DBIM_handbook_A_guide_to_buildi.

[89] Philipp Eberspächer and Alexander Verl. "Realizing Energy Reduction of Machine Tools Through a Control-Integrated Consumption Graph-Based Optimization Method". In: *Procedia CIRP*. Forty Sixth CIRP Conference on Manufacturing Systems 2013 7 (Jan. 1, 2013), pp. 640–645. ISSN: 2212-8271. DOI: 10.1016/j.procir.2013.06.046. URL: http://www.sciencedirect.com/science/article/pii/S2212827113003156 (visited on June 7, 2020).

[90] Matthias Ehrgott and Xavier Gandibleux. "Multiobjective Combinatorial Optimization — Theory, Methodology, and Applications". In: *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Ed. by Matthias Ehrgott and Xavier Gandibleux. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2002, pp. 369–444. ISBN: 978-0-306-48107-9. DOI: 10.1007/0-306-48107-3_8. URL: https://doi.org/10.1007/0-306-48107-3_8 (visited on Mar. 8, 2020).

[91] Bryan Eisenhower, Zheng O'Neill, Satish Narayanan, Vladimir A. Fonoberov, and Igor Mezić. "A Methodology for Meta-Model Based Optimization in Building Energy Models". In: *Energy and Buildings* 47 (Apr. 1, 2012), pp. 292–301. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2011.12.001. URL: http://www.sciencedirect.com/science/article/pii/S0378778811005962 (visited on May 3, 2020).

[92] Zineb El Akkaoui, José-Norberto Mazón, Alejandro Vaisman, and Esteban Zimányi. "BPMN-Based Conceptual Modeling of ETL Processes". In: *Data Warehousing and Knowledge Discovery*. Ed. by Alfredo Cuzzocrea and Umeshwar Dayal. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, and Gerhard Weikum. Vol. 7448. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–14. ISBN: 978-3-642-32583-0. DOI: 10.1007/978-3-642-32584-7_1. URL: http://link.springer.com/10.1007/978-3-642-32584-7_1 (visited on May 4, 2020).

[93] Hilding Elmqvist, Fabien Gaucher, Sven Erik Mattson, and Francois Dupont. "State Machines in Modelica". In: *Proceedings of the 9th International MODELICA Conference*. 9th International MODELICA Conference, Munich, Germany. Munich, Germany, Sept. 3–5, 2012, pp. 37–46. DOI: 10.3384/ecp1207637. URL: http://www.ep.liu.se/ecp/article.asp?issue=076%26article=3 (visited on June 8, 2020).

[94] Hilding Elmqvist, Sven Erik Mattsson, and Martin Otter. "Modelica Extensions for Multi-Mode DAE Systems". In: *Proceedings of the 10th International Modelica Conference*. Lund, Sweden, Mar. 10–12, 2014, pp. 183–193. DOI: 10.3384/ecp14096183. URL: http://www.ep.liu.se/ecp/article.asp?issue=096%26article=19 (visited on June 8, 2020).

[95] Federal Ministry for Sustainability and Tourism, Republic of Austria. *Energie in Österreich 2018 - Zahlen, Daten, Fakten*. Vienna, Austria: Federal Ministry for Sustainability and Tourism, Republic of Austria, 2018, p. 42. URL: https://www.bmlrt.gv.at/energie-bergbau/energie/Zahlen--Daten--Fakten.html (visited on Mar. 21, 2020).

[96] Yishai A. Feldman, Lev Greenberg, and Eldad Palachi. "Simulating Rhapsody SysML Blocks in Hybrid Models with FMI". In: *Proceedings of the 10th International Modelica Conference*. The 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden. Lund, Sweden, Mar. 10–12, 2014, pp. 43–52. DOI: 10.3384/ecp1409643. URL: http://www.ep.liu.se/ecp/article.asp?issue=096%26article=4 (visited on Feb. 12, 2020).

[97] Carlos A. Felippa, K. C. Park, and Charbel Farhat. "Partitioned Analysis of Coupled Mechanical Systems". In: *Computer Methods in Applied Mechanics and Engineering*. Advances in Computational Methods for Fluid-Structure Interaction 190.24–25 (Mar. 2, 2001), pp. 3247–3270. ISSN: 0045-7825. DOI: 10.1016/S0045-7825(00)00391-1. URL: http://www.sciencedirect.com/science/article/pii/S0045782500003911 (visited on Oct. 28, 2016).

[98] Thomas A. Feo and Mauricio G. C. Resende. "Greedy Randomized Adaptive Search Procedures". In: *Journal of Global Optimization* 6.2 (Mar. 1, 1995), pp. 109–133. ISSN: 1573-2916. DOI: 10.1007/BF01096763. URL: https://doi.org/10.1007/BF01096763 (visited on Mar. 18, 2020).

[99] P. Fey, H.W. Carter, and P.A. Wilsey. "Parallel Synchronization of Continuous Time Discrete Event Simulators". In: *Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162)*. Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162). Bloomington, IL, USA: IEEE Comput. Soc, Aug. 1997, pp. 227–231. ISBN: 978-0-8186-8108-0. DOI: 10.1109/ICPP.1997.622649. URL: http://ieeexplore.ieee.org/document/622649 (visited on Feb. 12, 2020).

[100] Paul A. Fishwick. *Simulation Model Design and Execution: Building Digital Worlds*. 1st. USA: Prentice Hall PTR, 1995. 432 pp. ISBN: 978-0-13-098609-2.

[101] John Fitzgerald, Ken Pierce, and Peter Gorm Larsen. "Co-Modelling and Co-Simulation in the Engineering of Systems of Cyber-Physical Systems". In: *2014 9th International Conference on System of Systems Engineering (SOSE)*. 2014 9th International Conference on System of Systems Engineering (SOSE). June 2014, pp. 67–72. DOI: 10.1109/SYSOSE.2014.6892465.

[102] Alex A. Freitas. "A Critical Review of Multi-Objective Optimization in Data Mining: A Position Paper". In: *ACM SIGKDD Explorations Newsletter* 6.2 (Dec. 1, 2004), pp. 77–86. ISSN: 1931-0145. DOI: 10.1145/1046456.1046467. URL: https://doi.org/10.1145/1046456.1046467 (visited on Mar. 4, 2020).

[103] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann, Oct. 23, 2014. 631 pp. ISBN: 978-0-12-800800-3. Google Books: Ze60AwAAQBAJ.

[104] Peter Fritzson. *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. Piscataway, NJ: IEEE Press, 2011. 211 pp. ISBN: 978-1-118-09424-2.

[105] Peter 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.

[106] Margaretha Gansterer, Christian Almeder, and Richard F. Hartl. "Simulation-Based Optimization Methods for Setting Production Planning Parameters". In: *International Journal of Production Economics* 151 (May 1, 2014). ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2013.10.016. URL: http://www.sciencedirect.com/science/article/pii/S092552731300457X (visited on Mar. 8, 2020).

[107] Alfredo Garro and Alberto Falcone. "On the Integration of HLA and FMI for Supporting Interoperability and Reusability in Distributed Simulation". In: *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*. DEVS '15. Alexandria, Virginia: Society for Computer Simulation International, Apr. 12, 2015, pp. 9–16. ISBN: 978-1-5108-0105-9.

[108] Michel Gendreau and Jean-Yves Potvin, eds. *Handbook of Metaheuristics*. Vol. 146. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2010. ISBN: 978-1-4419-1663-1. DOI: 10.1007/978-1-4419-1665-5. URL: https://doi.org/10.1007/978-1-4419-1665-5 (visited on Mar. 18, 2020).

[109] Luiza Gheorghe. "Continuous/Discrete Co-Simulation Interfaces from Formalization to Implementation". Dissertation. École Polytechnique de Montréal, 2009. URL: http://publications.polymtl.ca/137/ (visited on Oct. 24, 2016).

[110] Barbara Glock, Niki Popper, and Felix Breitenecker. "Various Aspects of Multi-Method Modelling and Its Applications in Modelling Large Infrastructure Systems Like Airports". In: *Proceedings of the European Modeling and Simulation Symposium*. 2015, p. 10. ISBN: 978-88-97999-57-7.

[111] Christian Goldmann. “Bestandsaufnahme der Funktionalität und Einsatzmöglichkeiten für APS-Systeme”. Diploma Thesis. Lüneburg, Germany: Fachhochschule Nordostniedersachsen, 2004.

[112] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. “Co-Simulation: A Survey”. In: *ACM Computing Surveys (CSUR)* 51.3 (May 23, 2018), 49:1–49:33. ISSN: 0360-0300. DOI: 10.1145/3179993. URL: https://doi.org/10.1145/3179993 (visited on Feb. 3, 2020).

[113] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. “Co-Simulation: State of the Art”. In: (2017). arXiv: 1702.00686. URL: http://arxiv.org/abs/1702.00686 (visited on Feb. 3, 2020).

[114] Francisco González, Manuel González, and Javier 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. URL: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1649579 (visited on Oct. 29, 2016).

[115] Abhijit Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Springer, 2015.

[116] Aljoscha Gruler, Javier Panadero, Jesica de Armas, José A. Moreno Pérez, and Angel A. Juan. “A Variable Neighborhood Search Simheuristic for the Multiperiod Inventory Routing Problem with Stochastic Demands”. In: *International Transactions in Operational Research* 27.1 (Jan. 1, 2020), pp. 314–335. ISSN: 0969-6016. DOI: 10.1111/itor.12540. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/itor.12540 (visited on Mar. 31, 2020).

[117] Bei Gu. “Co-Simulation of Algebraically Coupled Dynamic Subsytems”. Dissertation. Massachusetts Institute of Technology, Sept. 2001. URL: http://dspace.mit.edu/bitstream/handle/1721.1/8695/49837078-MIT.pdf?sequence=2 (visited on Oct. 29, 2016).

[118] Giancarlo Guizzardi and Gerd Wagner. “Conceptual Simulation Modeling with Onto-UML Advanced Tutorial”. In: *Proceedings of the 2012 Winter Simulation Conference (WSC)*. Proceedings of the 2012 Winter Simulation Conference (WSC). Dec. 2012, pp. 1–15. DOI: 10.1109/WSC.2012.6465133.

[119] M. Güller, Y. Uygun, and B. Noche. “Simulation-Based Optimization for a Capacitated Multi-Echelon Production-Inventory System”. In: *Journal of Simulation* 9.4 (Nov. 1, 2015), pp. 325–336. ISSN: 1747-7778. DOI: 10.1057/jos.2015.5. URL: https://doi.org/10.1057/jos.2015.5 (visited on Mar. 6, 2020).

[120] Holger Haag. *Eine Methodik zur modellbasierten Planung und Bewertung der Energieeffizienz in der Produktion*. Stuttgarter Beiträge zur Produktionsforschung 11. Stuttgart: Fraunhofer Verlag, 2013. ISBN: 978-3-8396-0547-9. URL: http://dx.doi.org/10.18419/opus-4536 (visited on Feb. 28, 2020).

[121] Holger Haag, Jörg Siegert, Thomas Bauernhansl, and Engelbert Westkämper. "An Approach for the Planning and Optimization of Energy Consumption in Factories Considering the Peripheral Systems". In: *Proceedings of the 19th CIRP International Conference on Life Cycle Engineering*. Ed. by David A. Dornfeld and Barbara S. Linke. Berkeley: Springer, 2012, pp. 335–339. ISBN: 978-3-642-29069-5. DOI: 10.1007/978-3-642-29069-5_57.

[122] Matthias Hacksteiner, Fabian Duer, Iman Ayatollahi, and Friedrich Bleicher. "Automatic Assessment of Machine Tool Energy Efficiency and Productivity". In: *Procedia CIRP*. 10th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '16. [Edited by: Roberto Teti, Manager Editor: Doriana M. D'Addona] 62 (Jan. 1, 2017), pp. 317–322. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.06.034. URL: http://www.sciencedirect.com/science/article/pii/S2212827116306527 (visited on June 7, 2020).

[123] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics 8. Springer Science & Business Media, Apr. 16, 2008. 540 pp. ISBN: 978-3-540-56670-0. Google Books: F93u7VcSRyYC.

[124] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics 14. Springer Science & Business Media, Mar. 14, 2013. 615 pp. ISBN: 978-3-662-09947-6. Google Books: NGvrCAAAQBAJ.

[125] Ahmad T. Al-Hammouri. "A Comprehensive Co-Simulation Platform for Cyber-Physical Systems". In: *Computer Communications* 36.1 (Dec. 1, 2012), pp. 8–19. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2012.01.003.

[126] Pierre Hansen and Nenad Mladenović. "Variable Neighborhood Search Methods". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 3975–3989. ISBN: 978-0-387-74758-3. DOI: 10.1007/978-0-387-74759-0_694. URL: http://link.springer.com/10.1007/978-0-387-74759-0_694 (visited on Mar. 4, 2020).

[127] Pierre Hansen, Nenad Mladenović, and José A. Moreno Pérez. "Variable Neighbourhood Search: Methods and Applications". In: *Annals of Operations Research* 175.1 (Mar. 1, 2010), pp. 367–407. ISSN: 1572-9338. DOI: 10.1007/s10479-009-0657-6. URL: https://doi.org/10.1007/s10479-009-0657-6 (visited on Mar. 18, 2020).

[128] C. Hardebolle and F. Boulanger. "Multi-Formalism Modelling and Model Execution". In: *International Journal of Computers and Applications* 31.3 (Jan. 1, 2009), pp. 193–203. ISSN: 1206-212X. DOI: 10.1080/1206212X.2009.11441941. URL: https://doi.org/10.1080/1206212X.2009.11441941 (visited on Feb. 5, 2020).

[129] Berhard Heinzl and Wolfgang Kastner. “A General Variable Neighborhood Search for Simulation-Based Energy-Aware Flow Shop Scheduling”. In: *Proceedings of the 2020 Summer Simulation Conference*. Virtual Event: SCS, July 20–22, 2020.

[130] Bernhard Heinzl. “Hybrid Modeling of Production Systems: Co-Simulation and DEVS-Based Approach”. Diploma Thesis. Vienna, Austria: TU Wien, Nov. 2016. URL: http://katalog.ub.tuwien.ac.at/AC13372722 (visited on Mar. 11, 2018).

[131] Bernhard Heinzl. “Objektorientierte Multi-Domain-Modellierung und Simulation von Werkzeugmaschinen”. Diploma Thesis. Wien: TU Wien, June 2012.

[132] Bernhard Heinzl and Wolfgang Kastner. “Platform-Independent Modeling for Simulation-Based Energy Optimization in Industrial Production”. In: *International Journal of Simulation: Systems, Science and Technology* 20.6 (Dec. 2019), pp. 10.1–10.10. DOI: 10.5013/IJSSST.a.20.06.10.

[133] Bernhard Heinzl and Wolfgang Kastner. “Towards Model-Driven Development of Hybrid Simulation Models in Industrial Engineering”. In: *Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society (IECON 2018)*. Washington, D.C., USA: IEEE, Oct. 2018, pp. 3588–3593. DOI: 10.1109/IECON.2018.8592811. URL: https://ieeexplore.ieee.org/abstract/document/8592811 (visited on Jan. 10, 2019).

[134] Bernhard Heinzl, Wolfgang Kastner, Ines Leobner, Fabian Dür, Friedrich Bleicher, and Iva Kovacic. “Using Coupled Simulation for Planning of Energy Efficient Production Facilities”. In: *Proceedings of the 2014 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. Berlin, Germany: IEEE, Apr. 2014, pp. 21–26. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6842397 (visited on Oct. 9, 2016).

[135] Bernhard Heinzl, Philipp Raich, Franz Preyser, and Wolfgang Kastner. “Simulation-Based Assessment of Energy Efficiency in Industry: Comparison of Hybrid Simulation Approaches”. In: *Proceedings of the 9th Vienna International Conference on Mathematical Modelling (MATHMOD 2018)*. Wien, Austria, Feb. 2018.

[136] Bernhard Heinzl, Philipp Raich, Franz Preyser, Wolfgang Kastner, Peter Smolek, and Ines Leobner. “Modular Hybrid Modeling Based on DEVS for Interdisciplinary Simulation of Production Systems”. In: *Proceedings of the 31st European Simulation and Modelling Conference (ESM 2017)*. Lisbon, Portugal, Oct. 2017, pp. 157–161. ISBN: 978-94-92859-00-6.

[137] Bernhard Heinzl, Matthias Rößler, Nikolas Popper, Ines Leobner, Karl Ponweiser, Wolfgang Kastner, Fabian Dür, Friedrich Bleicher, and Felix Breitenecker. “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. URL: http://ieeexplore.ieee.org/abstract/document/6527433/ (visited on Mar. 11, 2018).

[138] C. Herrmann, S. Thiede, S. Kara, and J. Hesselbach. “Energy Oriented Simulation of Manufacturing Systems - Concept and Application”. In: *CIRP Annals - Manufacturing Technology* 60.1 (2011), pp. 45–48. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2011.03.127. URL: http://www.sciencedirect.com/science/article/pii/S0007850611001284 (visited on Jan. 29, 2020).

[139] Christoph Herrmann and Sebastian Thiede. “Process Chain Simulation to Foster Energy Efficiency in Manufacturing”. In: *CIRP Journal of Manufacturing Science and Technology* 1.4 (2009), pp. 221–229. ISSN: 17555817. DOI: 10.1016/j.cirpj.2009.06.005. URL: http://www.sciencedirect.com/science/article/pii/S1755581709000121.

[140] Jens Hesselbach. *Energie- und klimaeffiziente Produktion: Grundlagen, Leitlinien und Praxisbeispiele*. Springer-Verlag, 2012. 348 pp. ISBN: 978-3-8348-0448-8. URL: https://doi.org/10.1007/978-3-8348-9956-9.

[141] Edward Huang, Randeep Ramamurthy, and Leon F. McGinnis. “System and Simulation Modeling Using {SysML”. In: *2007 Winter Simulation Conference*. 2007 Winter Simulation Conference. Dec. 2007, pp. 796–803. DOI: 10.1109/WSC.2007.4419675.

[142] Philipp Huber. “The Model Transformation Language Jungle - An Evaluation and Extension of Existing Approaches”. Master Thesis. Vienna, Austria: TU Wien, 2008. URL: http://katalog.ub.tuwien.ac.at/AC05037690 (visited on Apr. 21, 2020).

[143] C.-L. Hwang and A. S. M. Masud. *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey*. Springer Science & Business Media, Dec. 6, 2012. 366 pp. ISBN: 978-3-642-45511-7. Google Books: M0noCAAAQBAJ.

[144] IEC 62264-1:2013. *Enterprise-Control System Integration – Part 1: Models and Terminology*. 2013. URL: https://www.iso.org/standard/57308.html.

[145] 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. URL: http://ieeexplore.ieee.org/servlet/opac?punumber=5553438 (visited on Nov. 10, 2016).

[146] M.D. Ilic, Le Xie, U.A. Khan, and Jose M.F. Moura. “Modeling Future Cyber-Physical Energy Systems”. In: *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*. 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century. July 2008, pp. 1–9. DOI: 10.1109/PES.2008.4596708.

[147] Frank Iwanitz and Jürgen Lange. *OPC: Grundlagen, Implementierung und Anwendung*. Hüthig, 2005. 271 pp. ISBN: 978-3-7785-2903-4. Google Books: agNxAAAACAAJ.

[148] Raymond G Jacquot. *Modern Digital Control Systems*. CRC Press, 2018. ISBN: 978-1-351-43057-9.

[149] Adithya Jayakumar. "Simulation-Based Optimization of Hybrid Systems Using Derivative Free Optimization Techniques". The Ohio State University, 2018. URL: https://etd.ohiolink.edu/pg_10?0::NO:10:P10_ACCESSION_NUM:osu1531954151797307 (visited on Mar. 31, 2020).

[150] Sabina Jeschke, Christian Brecher, Tobias Meisen, Denis Özdemir, and Tim Eschert. "Industrial Internet of Things and Cyber Manufacturing Systems". In: *Industrial Internet of Things: Cybermanufacturing Systems*. Ed. by Sabina Jeschke, Christian Brecher, Houbing Song, and Danda B. Rawat. Springer Series in Wireless Technology. Cham: Springer International Publishing, 2017, pp. 3–19. ISBN: 978-3-319-42559-7. DOI: 10.1007/978-3-319-42559-7_1. URL: https://doi.org/10.1007/978-3-319-42559-7_1 (visited on June 3, 2020).

[151] Angel A. Juan, Javier Faulin, Scott E. Grasman, Markus Rabe, and Gonçalo Figueira. "A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Combinatorial Optimization Problems". In: *Operations Research Perspectives* 2 (Dec. 1, 2015), pp. 62–72. ISSN: 2214-7160. DOI: 10.1016/j.orp.2015.03.001. URL: http://www.sciencedirect.com/science/article/pii/S221471601500007X (visited on Mar. 31, 2020).

[152] Mark Junge. "Simulationsgestützte Entwicklung Und Optimierung Einer Energieeffizienten Produktionssteuerung". Dissertation. Deutschland: Universität Kassel, 2007. 170 pp.

[153] Henning Kagermann, Johannes Helbig, Ariane Hellinger, and Wolfgang Wahlster. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*. Platform Industrie 4.0, Apr. 2013. URL: https://www.din.de/blob/76902/e8cac883f42bf28536e7e8165993f1fd/recommendations-for-implementing-industry-4-0-data.pdf.

[154] Felix Kamhuber, Thomas Sobottka, Bernhard Heinzl, and Wilfried Sihn. "An Efficient Multi-Objective Hybrid Simheuristic Approach for Advanced Rolling Horizon Production Planning". In: *Proceedings of the 2019 Winter Simulation Conference*. 2019 Winter Simulation Conference. Vol. 1. Maryland, USA: IEEE, Dec. 8–11, 2019, pp. 1–11.

[155] Hyoung Seok Kang, Ju Yeon Lee, SangSu Choi, Hyun Kim, Jun Hee Park, Ji Yeon Son, Bo Hyun Kim, and Sang Do Noh. "Smart Manufacturing: Past Research, Present Findings, and Future Directions". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 3.1 (2016), pp. 111–128. ISSN: 2198-0810. DOI: 10.1007/s40684-016-0015-5. URL: https://doi.org/10.1007/s40684-016-0015-5 (visited on Mar. 1, 2020).

[156] M. Karimi-Nasab and M.B. Aryanezhad. “A Multi-Objective Production Smoothing Model with Compressible Operating Times”. In: *Applied Mathematical Modelling* 35.7 (July 2011), pp. 3596–3610. ISSN: 0307-904X. DOI: 10.1016/j.apm.2011.01.038. URL: http://www.sciencedirect.com/science/article/pii/S0307904X11000527 (visited on Mar. 8, 2020).

[157] A. Kaya. “Improving Efficiency in Existing Chillers with Optimization Technology”. In: *ASHRAE Journal (American Society of Heating, Refrigerating and Air-Conditioning Engineers)* 33:10 (Oct. 1, 1991). ISSN: 0001-2491. URL: https://www.osti.gov/biblio/5639276 (visited on Mar. 31, 2020).

[158] Florian Kellner, Bernhard Lienland, and Maximilian Lukesch. “Produktionsplanung Und -Steuerung (PPS)”. In: *Produktionswirtschaft : Planung, Steuerung Und Industrie 4.0*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 131–272. ISBN: 978-3-662-54341-2. DOI: 10.1007/978-3-662-54341-2âĆČ. URL: https://doi.org/10.1007/978-3-662-54341-2%E2%82%83.

[159] K. Khedri Liraviasl, H. ElMaraghy, M. Hanafy, and S. N. Samy. “A Framework for Modelling Reconfigurable Manufacturing Systems Using Hybridized Discrete-Event and Agent-Based Simulation”. In: *IFAC-PapersOnLine*. 15th IFAC Symposium on-Information Control Problems inManufacturing 48.3 (Jan. 1, 2015), pp. 1490–1495. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.06.297. URL: http://www.sciencedirect.com/science/article/pii/S2405896315005364 (visited on Feb. 4, 2020).

[160] Rüdiger Kiesel and Florentina Paraschiv. “Econometric Analysis of 15-Minute Intraday Electricity Prices”. In: *Energy Economics* 64 (May 1, 2017), pp. 77–90. ISSN: 0140-9883. DOI: 10.1016/j.eneco.2017.03.002. URL: http://www.sciencedirect.com/science/article/pii/S0140988317300683 (visited on Mar. 31, 2020).

[161] Christoph Kilger, Herbert Meyr, and Hartmut Stadtler. *Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies*. Springer, 2015.

[162] Cheol-Han Kim, R. H. Weston, A. Hodgson, and Kyung-Huy Lee. “The Complementary Use of IDEF and UML Modelling Approaches”. In: *Computers in Industry* 50.1 (Jan. 1, 2003), pp. 35–56. ISSN: 0166-3615. DOI: 10.1016/S0166-3615(02)00145-8. URL: http://www.sciencedirect.com/science/article/pii/S0166361502001458 (visited on May 4, 2020).

[163] Kihyung Kim, Wonseok Kang, Bong Sagong, and Hyungon 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. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=844920 (visited on Oct. 30, 2016).

[164] Jürgen Kletti. *Konzeption und Einführung von MES-Systemen: zielorientierte Einführungsstrategie mit Wirtschaftlichkeitsbetrachtungen, Fallbeispielen und Checklisten*. Berlin: Springer, 2007. 294 pp. ISBN: 978-3-540-34309-7.

[165] E. Kofman. "Relative Error Control in Quantization Based Integration". In: *Latin American applied research* 39.3 (July 2009), pp. 231–237. ISSN: 0327-0793. URL: http://bibliotecadigital.uns.edu.ar/scielo.php?script=sci_abstract&pid=S0327-07932009003300009&lng=pt&nrm=iso&tlng=en (visited on May 2, 2020).

[166] Ernesto Kofman and Sergio Junco. "Quantized-State Systems: A DEVS Approach for Continuous System Simulation". In: *Transactions of The Society for Modeling and Simulation International* 18.3 (2001). URL: https://www.fceia.unr.edu.ar/~kofman/files/qss.pdf (visited on Oct. 30, 2016).

[167] Martin Koller and René Hofmann. "Dynamic and Predictive Optimization Concept for Energy Supply Systems in the Energy-Intensive Industry". In: ASME 2016 International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers Digital Collection, Feb. 8, 2017. DOI: 10.1115/IMECE2016-66536. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2016/50657/V011T15A024/263491 (visited on Mar. 29, 2020).

[168] Konsortium Project Balanced Manufacturing. *Documentation of BaMa Methodology*. 2015. URL: http://bama.ift.tuwien.ac.at/fileadmin/t/bama/Documentation_of_BaMa_Methodology.pdf (visited on Oct. 10, 2016).

[169] Konsortium Project INFO. *{INFO} - Interdisziplinäre Forschung zur Energieoptimierung in Fertigungsbetrieben: Endbericht*. Publizierbarer Endbericht. Wien: Konsortium Project INFO, Sept. 30, 2013, p. 132. URL: http://www.projekt-info.org/endbericht/2013-09-30%20publizierbarer_endbericht_final.pdf (visited on Oct. 9, 2016).

[170] Kathy Kotiadis and Stewart Robinson. "Conceptual Modelling: Knowledge Acquisition and Model Abstraction". In: *2008 Winter Simulation Conference*. 2008 Winter Simulation Conference. Dec. 2008, pp. 951–958. DOI: 10.1109/WSC.2008.4736161.

[171] Velin Kounev, David Tipper, Martin Levesque, Brandon M. Grainger, Thomas Mcdermott, and Gregory F. Reed. "A Microgrid Co-Simulation Framework". In: *2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES). Seattle, WA, USA: IEEE, Apr. 2015, pp. 1–6. DOI: 10.1109/MSCPES.2015.7115398.

[172] Iva Kovacic, Kristina Orehounig, Ardeshir Mahdavi, Friedrich Bleicher, Alexandros-Athanassios Dimitrou, and Linus Waltenbereger. "Energy Efficient Production – Interdisciplinary, Systemic Approach through Integrated Simulation". In: *Strojarstvo: Journal for Theory and Application in Mechanical Engineering* 55.1 (2013),

pp. 17–34. ISSN: 0562-1887. URL: https://publik.tuwien.ac.at/files/PubDat_222974.pdf.

[173] Klaus Krämer. *Automatisierung in Materialfluss und Logistik: Ebenen, Informationslogistik, Identifikationssysteme, intelligente Geräte*. Springer, Mar. 7, 2013. 266 pp. ISBN: 978-3-322-81221-6. Google Books: p4r0BQAAQBAJ.

[174] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. "Digital Twin in Manufacturing: A Categorical Literature Review and Classification". In: *IFAC-PapersOnLine*. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018 51.11 (Jan. 1, 2018), pp. 1016–1022. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.08.474. URL: http://www.sciencedirect.com/science/article/pii/S2405896318316021 (visited on June 8, 2020).

[175] Thomas Kuehne and Daniel Schreiber. "Can Programming Be Liberated from the Two-Level Style: Multi-Level Programming with Deepjava". In: *Proceedings of the 22Nd Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications*. OOPSLA '07. New York, NY, USA: ACM, 2007, pp. 229–244. ISBN: 978-1-59593-786-5. DOI: 10.1145/1297027.1297044. URL: http://doi.acm.org/10.1145/1297027.1297044 (visited on Feb. 16, 2018).

[176] Thomas Kühne. "Matters of (Meta-) Modeling". In: *Software & Systems Modeling* 5.4 (2006), pp. 369–385.

[177] Jens Kuhpfahl. *Job Shop Scheduling with Consideration of Due Dates: Potentials of Local Search Based Solution Techniques*. Springer, June 24, 2015. 206 pp. ISBN: 978-3-658-10292-0.

[178] Thomas Kuhr, Thomas Forster, Tobias Braun, and Reinhard Gotzhein. "FERAL – Framework for Simulator Coupling on Requirements and Architecture Level". In: *2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013)*. 2013 Eleventh ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2013). Oct. 2013, pp. 11–22.

[179] Andrew Kusiak. "Smart Manufacturing". In: *International Journal of Production Research* 56.1-2 (Jan. 17, 2018), pp. 508–517. ISSN: 0020-7543. DOI: 10.1080/00207543.2017.1351644. URL: https://doi.org/10.1080/00207543.2017.1351644 (visited on Mar. 1, 2020).

[180] Juan De Lara, Esther Guerra, and Jesús Sánchez Cuadrado. "When and How to Use Multilevel Modelling". In: *ACM Transactions on Software Engineering and Methodology* 24.2 (Dec. 23, 2014), 12:1–12:46. ISSN: 1049-331X. DOI: 10.1145/2685615. URL: https://doi.org/10.1145/2685615 (visited on May 11, 2020).

[181] Averill M. Law. *Simulation Modeling and Analysis*. 5. ed. McGraw-Hill Series in Industrial Engineering and Management Science. New York, NY: McGraw-Hill Education, 2015. 776 pp. ISBN: 978-0-07-340132-4.

[182] Edward A. Lee and Haiyang Zheng. "Operational Semantics of Hybrid Systems". In: *Hybrid Systems: Computation and Control*. Ed. by Manfred Morari and Lothar Thiele. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 25–53. ISBN: 978-3-540-31954-2. DOI: 10.1007/978-3-540-31954-2_2.

[183] Ines Leobner. "Modeling of Energy Systems for Complex Simulations". Dissertation. Wien: TU Wien, 2016. URL: http://publik.tuwien.ac.at/files/PubDat_247849.pdf (visited on Oct. 31, 2016).

[184] Ines Leobner, Wolfgang Kastner, Iva Kovacic, Bernhard Heinzl, Matthias Rößler, Fabian Dür, Thomas Flatz, Karl Ponweiser, and Friedrich Bleicher. "Interdisziplinäre Optimierungsstrategien Zu Energieeffizienz in Produktionsbetrieben". In: *8. Internationale Energiewirtschaftstagung*. Wien, Feb. 2013, pp. 1–11.

[185] Ines Leobner, Peter Smolek, Bernhard Heinzl, Philipp Raich, Alexander Schirrer, Martin Kozek, Matthias Rössler, and Benjamin Mörzinger. "Simulation-Based Strategies for Smart Demand Response". In: *Journal of Sustainable Development of Energy, Water and Environment Systems* 6.1 (Mar. 2018), pp. 33–46. ISSN: 18489257. DOI: 10.13044/j.sdewes.d5.0168. URL: http://www.sdewes.org/jsdewes/pid5.0168 (visited on Mar. 9, 2018).

[186] Wen Li. *Efficiency of Manufacturing Processes*. Sustainable Production, Life Cycle Engineering and Management. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-17364-1. DOI: 10.1007/978-3-319-17365-8. URL: http://link.springer.com/10.1007/978-3-319-17365-8 (visited on Feb. 20, 2020).

[187] Hua Lin, Santhoshkumar Sambamoorthy, Sandeep Shukla, James Thorp, and Lamine Mili. "Power System and Communication Network Co-Simulation for Smart Grid Applications". In: *IEEE PES Innovative Smart Grid Technologies Conference Europe, Isgt Europe*. IEEE, 2011, pp. 1–6. ISBN: 978-1-61284-218-9. DOI: 10.1109/ISGT.2011.5759166. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5759166.

[188] John Lygeros. "Lecture Notes on Hybrid Systems". In: *Notes for an ENSIETA Workshop*. 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.644&rep=rep1&type=pdf.

[189] Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm. *OPC Unified Architecture*. Springer Science & Business Media, Apr. 5, 2009. 350 pp. ISBN: 978-3-540-68899-0. Google Books: de9uLdXKj1IC.

[190] Lars Martin, Jens Hesselbach, Sebastian Thiede, Christoph Herrmann, Bruno Lüdemann, and Rudiger Detzer. "Energieeffizienz durch optimierte Abstimmung zwischen Produktion und technischer Gebäudeausrüstung". In: *Proceedings of the 13. Fachtagung Simulation in Produktion und Logistik (Advances in Simulation for Production and Logistics Applications) ASIM SPL 2008*. Berlin: Fraunhofer IRB Verlag, Stuttgart, Oct. 1–2, 2008, p. 9. ISBN: 978-3-8167-7798-4.

[191] Marcellus Menges, Daniil Roubanov, and Joscha Ernst. “Produktionsplanung Und -Steuerung (PPS)”. In: *Modellbasierte Virtuelle Produktentwicklung*. Ed. by Martin Eigner, Daniil Roubanov, and Radoslav Zafirov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 301–325. ISBN: 978-3-662-43816-9. URL: https://doi.org/10.1007/978-3-662-43816-9%E2%82%813.

[192] Tobias Meudt, Malte Pohl, and Joachim Metternich. *Die Automatisierungspyramide - Ein Literaturüberblick*. Germany: TU Darmstadt, 2017. URL: http://tuprints.ulb.tu-darmstadt.de/6298/1/2017%20-%20Die%20Automatisierungspyramide%20-%20Ein%20Literatur%C3%BCberblick-2.pdf.

[193] J.A. Miller, G.T. Baramidze, A.P. Sheth, and P.A. Fishwick. “Investigating Ontologies for Simulation Modeling”. In: *Proceedings of the 37th Annual Simulation Symposium (ANSS'04)*. 37th Annual Simulation Symposium (ANSS'04). Apr. 2004, pp. 55–63. DOI: 10.1109/SIMSYM.2004.1299465.

[194] Dr R. J. Mitchell. *Managing Complexity in Software Engineering*. Vol. 17. IET, 1990. 284 pp. ISBN: 978-0-86341-171-7. Google Books: uXtHeZt8ZowC.

[195] Saurabh Mittal and José L. Risco Martín. *Netcentric System of Systems Engineering with DEVS Unified Process*. CRC Press, Sept. 3, 2018. 715 pp. ISBN: 978-1-351-83374-5.

[196] Modelica Association. *Modelica - A Unified Object-Oriented Language for Systems Modeling: Language Specification Version 3.3 Revision 1*. July 11, 2014. URL: https://www.modelica.org/documents/%20ModelicaSpec33Revision1.pdf (visited on Oct. 8, 2016).

[197] Benjamin Mörzinger, Martin Obermair, Ines Leobner, Peter Smolek, Bernhard Heinzl, Andreas Wittmann, Matthias Rößler, and Friedrich Bleicher. “Modelling, Simulation and Optimization of the Operation of Electric Chillers”. In: *Proceedings of the 6th International Symposium on Energy Challenges and Mechanics - towards a Big Picture (ECM2016)*. Inverness, Scotland, UK, Aug. 2016. URL: http://nscj.co.uk/ecm6/sessions/A06_090.pdf (visited on Nov. 28, 2016).

[198] Ralf Mosshammer, Friederich Kupzog, Mario Faschang, and Matthias 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. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6700394 (visited on Oct. 6, 2016).

[199] Pierre-Alain Muller, Frédéric Fondement, Franck Fleurey, Michel Hassenforder, Rémi Schnekenburger, Sébastien Gérard, and Jean-Marc Jézéquel. “Model-Driven Analysis and Synthesis of Textual Concrete Syntax”. In: *Software & Systems Modeling* 7.4 (Oct. 1, 2008), pp. 423–441. ISSN: 1619-1374. DOI: 10.1007/s10270-008-0088-x. URL: https://doi.org/10.1007/s10270-008-0088-x (visited on May 6, 2020).

[200] Goran Music. “Automation Virtual Architecture: A Component-Oriented Abstraction Layer for "Virtual Plug&Produce" Automation Systems Engineering”. Diploma Thesis. Vienna, Austria: TU Wien, 2020.

[201] Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh, Hubertus Tummescheit, and Chandrasekar Sureshkumar. “Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems”. In: *Proceedings of the 10th International Modelica Conference*. Lund, Sweden, Mar. 10–12, 2014, pp. 235–245. DOI: 10.3384/ecp14096235. URL: http://www.ep.liu.se/ecp/article.asp?issue=096%26article=24 (visited on Feb. 12, 2020).

[202] Reimund Neugebauer, ed. *Handbuch Ressourcenorientierte Produktion*. München: Carl Hanser Verlag GmbH & Co. KG, Nov. 2013. ISBN: 978-3-446-43008-2. DOI: 10.3139/9783446436237. URL: http://www.hanser-elibrary.com/doi/book/10.3139/9783446436237 (visited on Feb. 21, 2020).

[203] Mara Nikolaidou, Vassilis Dalakas, Loreta Mitsi, George-Dimitrios Kapos, and Dimosthenis Anagnostopoulos. “A SysML Profile for Classical DEVS Simulators”. In: *The Third International Conference on Software Engineering Advances*. 2008, pp. 445–450. URL: http://galaxy.hua.gr/~dimosthe/publications/icsea08.pdf.

[204] James Nutaro. “Designing Power System Simulators for the Smart Grid: Combining Controls, Communications, and Electro-Mechanical Dynamics”. In: *2011 IEEE Power and Energy Society General Meeting*. 2011 IEEE Power & Energy Society General Meeting. San Diego, CA: IEEE, July 2011, pp. 1–5. ISBN: 978-1-4577-1000-1. DOI: 10.1109/PES.2011.6039456. URL: https://ieeexplore.ieee.org/document/6039456/ (visited on Feb. 12, 2020).

[205] Antoni Olivé. *Conceptual Modeling of Information Systems*. Springer Science & Business Media, Aug. 28, 2007. 471 pp. ISBN: 978-3-540-39389-4. Google Books: RoLZWcdA4VkC.

[206] OMG. *MDA Guide Version 1.0.1*. 2003. URL: https://www.omg.org/mda/specs.htm (visited on May 4, 2020).

[207] OMG. *Unified Modeling Language (UML) Infrastructure Version 2.5.1*. 2017. URL: https://www.omg.org/spec/UML/2.5.1.

[208] Martin Otter, Karl-Erik Årzén, and Isolde Dressler. “Stategraph – a Modelica Library for Hierarchical State Machines”. In: *Proceedings of the 4th International Modelica Conference*. Mar. 7–8, 2005, pp. 569–578. URL: https://elib.dlr.de/12300/.

[209] Dale K Pace. “Ideas About Simulation Conceptual Model Development”. In: *Johns Hopkins APL technical digest* 21.3 (2000), pp. 327–336.

[210] Peter Palensky and Dietmar Dietrich. “Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads”. In: *IEEE Transactions on Industrial Informatics* 7.3 (Aug. 2011), pp. 381–388. ISSN: 1941-0050. DOI: 10.1109/TII.2011.2158841.

[211] Murray G Patterson. “What Is Energy Efficiency?: Concepts, Indicators and Methodological Issues”. In: *Energy Policy* 24.5 (May 1, 1996), pp. 377–390. ISSN: 0301-4215. DOI: 10.1016/0301-4215(96)00017-1. URL: http://www.sciencedirect.com/science/article/pii/0301421596000171 (visited on Feb. 20, 2020).

[212] Thorsten Pawletta, Artur Schmidt, and Peter Junglas. “A Multimodeling Approach for the Simulation of Energy Consumption in Manufacturing”. In: *SNE Simulation Notes Europe* 27.2 (June 2017), pp. 115–124. ISSN: 2305-9974. DOI: 10.11128/sne.27.tn.10377. URL: https://www.sne-journal.org/10377 (visited on Jan. 29, 2020).

[213] Thorsten Pawletta, Artur Schmidt, Bernard P. Zeigler, and Umut Durak. “Extended Variability Modeling Using System Entity Structure Ontology Within Matlab/Simulink”. In: *Proceedings of the 49th Annual Simulation Symposium (SpringSim-ANSS 2016)*. 2016 Spring Simulation Multi-Conference. Pasadena, CA, USA: SCS, 2016. ISBN: 978-1-5108-2316-7. DOI: 10.22360/SpringSim.2016.ANSS.061. URL: http://dl.acm.org/citation.cfm?id=2962396 (visited on June 8, 2020).

[214] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. “A Design Science Research Methodology for Information Systems Research”. In: *Journal of Management Information Systems* 24.3 (Dec. 1, 2007), pp. 45–77. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302. URL: https://doi.org/10.2753/MIS0742-1222240302 (visited on June 8, 2020).

[215] Tim Peter and Sigrid Wenzel. “Simulationsgestützte Planung Und Bewertung Der Energieeffizienz Für Produktionssysteme in Der Automobilindustrie”. In: *Markus Rabe & Uwe Clausen (eds.), Simulation in Production and Logistics* (2015), pp. 535–544. URL: http://www.asim-fachtagung-spl.de/asim2015/papers/Proof_110_Wenzel-V2.pdf (visited on Oct. 16, 2017).

[216] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. USA: Prentice Hall, 1981. 288 pp. ISBN: 978-0-13-661983-3.

[217] Antonella Petrillo, Raffaele Cioffi, and Fabio De Felice. *Digital Transformation in Smart Manufacturing*. BoD – Books on Demand, Feb. 28, 2018. 158 pp. ISBN: 978-953-51-3841-9. Google Books: 2WaQDwAAQBAJ.

[218] Yves Pochet and Laurence A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 978-0-387-29959-4. DOI: 10.1007/0-387-33477-7. URL: http://link.springer.com/10.1007/0-387-33477-7 (visited on Mar. 18, 2020).

[219] Nikolas Popper. “Comparative Modelling and Simulation: A Concept for Modular Modelling and Hybrid Simulation of Complex Systems”. PhD Thesis. Vienna, Austria: TU Wien, 2015. URL: http://katalog.ub.tuwien.ac.at/AC12309065 (visited on May 13, 2020).

[220] Herbert Prähofer. “System Theoretic Formalisms for Combined Discrete-Continuous System Simulation”. In: *International Journal of General Systems* 19.3 (Oct. 1, 1991), pp. 219–240. ISSN: 0308-1079. DOI: 10.1080/03081079108935175. URL: https://doi.org/10.1080/03081079108935175 (visited on Oct. 30, 2016).

[221] Herbert Prähofer. “System Theoretic Foundations for Combined Discrete-Continuous System Simulation”. Dissertation. Linz, Austria: Johannes Kepler Universität Linz, 1991.

[222] Franz Preyser. “An Approach to Develop a User Friendly Way of Implementing DEV&DESS Models in PowerDEVS”. Diploma Thesis. Wien: TU Wien, 2015. URL: http://katalog.ub.tuwien.ac.at/AC12315517 (visited on Oct. 29, 2016).

[223] ARNO Puder, KAY Römer, and FRANK Pilhofer. “CORBA and Beyond”. In: *Distributed Systems Architecture*. Ed. by ARNO Puder, KAY Römer, and FRANK Pilhofer. San Francisco: Morgan Kaufmann, Jan. 1, 2006, pp. 183–229. ISBN: 978-1-55860-648-7. DOI: 10.1016/B978-155860648-7/50011-5. URL: http://www.sciencedirect.com/science/article/pii/B9781558606487500115 (visited on May 11, 2020).

[224] Clemens Pühringer. *Using the Modelica Language to Simulate Hybrid Models*. Student Project. Wien: TU Wien, Oct. 3, 2016.

[225] Davide Quaglia, Riccardo Muradore, Roberto Bragantini, and Paolo Fiorini. “A SystemC/Matlab Co-Simulation Tool for Networked Control Systems”. In: *Simulation Modelling Practice and Theory* 23 (Apr. 1, 2012), pp. 71–86. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2012.01.003. URL: http://www.sciencedirect.com/science/article/pii/S1569190X12000111 (visited on Feb. 12, 2020).

[226] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer Science & Business Media, Nov. 30, 2010. 664 pp. ISBN: 978-3-540-49809-4.

[227] Gauthier Quesnel, Raphaël Duboz, David Versmisse, and Éric Ramat. “DEVS Coupling of Spatial and Ordinary Differential Equations: VLE Framework”. In: *Proceedings of the 1st Open International Conference on Modeling and Simulation (OICMS 2005)*. Clermont-Ferrand, France, June 13–15, 2005, p. 14. URL: https://pdfs.semanticscholar.org/2b83/e87868ec2652a341cc1e6f71dc04fcf6d8a7.pdf.

[228] Markus Rager. *Energieorientierte Produktionsplanung: Analyse, Konzeption und Umsetzung*. Springer-Verlag, Aug. 17, 2008. 154 pp. ISBN: 978-3-8350-5569-8. Google Books: 9j2EYZ0159EC.

[229] Philipp Raich, Bernhard Heinzl, Franz Preyser, and Wolfgang Kastner. "Modeling Techniques for Integrated Simulation of Industrial Systems Based on Hybrid PDEVS". In: *Proceedings of the 2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. Vienna, Austria: IEEE, Apr. 2016, pp. 1–6. ISBN: 978-1-5090-1158-2. DOI: 10.1109/MSCPES.2016.7480221. URL: http://ieeexplore.ieee.org/document/7480221/?arnumber=7480221%20http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7480221 (visited on Mar. 11, 2018).

[230] S Robinson. "Conceptual Modelling for Simulation Part I: Definition and Requirements". In: *Journal of the Operational Research Society* 59.3 (Mar. 1, 2008), pp. 278–290. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2602368. URL: https://orsociety.tandfonline.com/doi/full/10.1057/palgrave.jors.2602368 (visited on May 4, 2020).

[231] S. Robinson. "Conceptual Modelling for Simulation Part II: A Framework for Conceptual Modelling". In: *Journal of the Operational Research Society* 59.3 (Mar. 1, 2008), pp. 291–304. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2602369. URL: https://doi.org/10.1057/palgrave.jors.2602369 (visited on May 4, 2020).

[232] Stewart Robinson. "Conceptual Modeling for Simulation: Issues and Research Requirements". In: *Proceedings of the 2006 Winter Simulation Conference (WSC'06)*. 2006 Winter Simulation Conference. Dec. 2006, pp. 792–800. DOI: 10.1109/WSC.2006.323160.

[233] Stewart Robinson, Roger Brooks, Kathy Kotiadis, and Durk-Jouke Van Der Zee. *Conceptual Modeling for Discrete-Event Simulation*. CRC Press, Aug. 2, 2010. 530 pp. ISBN: 978-1-4398-1038-5. Google Books: Z5rLBQAAQBAJ.

[234] Anna Carina Römer, Martina Rückbrod, and Steffen Straßburger. "Eignung kombinierter Simulation zur Darstellung energetischer Aspekte in der Produktionssimulation". In: *24. Symposium Simulationstechnik ASIM 2018*. Hamburg, Germany, Oct. 4–5, 2018, pp. 73–80.

[235] Anna Carina Römer and Steffen Strassburger. "A Review of Literature on Simulation-Based Optimization of the Energy Efficiency in Production". In: *Proceedings of the 2016 Winter Simulation Conference (WSC)*. 2016 Winter Simulation Conference (WSC). IEEE, Dec. 2016, pp. 1416–1427. DOI: 10.1109/WSC.2016.7822194. URL: http://ieeexplore.ieee.org/abstract/document/7822194/ (visited on Apr. 28, 2017).

[236] Anna Carina Römer and Steffen Strassburger. "Hybrid System Modeling Approach for the Depiction of the Energy Consumption in Production Simulations". In: *Proceedings of the 2019 Winter Simulation Conference (WSC)*. Dec. 2019, p. 12.

[237] Stefan Ropke and David Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* (Nov. 1, 2006). DOI: 10.1287/trsc.1050.0135. URL: https://pubsonline.informs.org/doi/abs/10.1287/trsc.1050.0135 (visited on Mar. 28, 2020).

[238] V. Roshanaei, B. Naderi, F. Jolai, and M. Khalili. "A Variable Neighborhood Search for Job Shop Scheduling with Set-up Times to Minimize Makespan". In: *Future Generation Computer Systems* 25.6 (June 1, 2009), pp. 654–661. ISSN: 0167-739X. DOI: 10.1016/j.future.2009.01.004. URL: http://www.sciencedirect.com/science/article/pii/S0167739X09000028 (visited on Mar. 8, 2020).

[239] Franz Rothlauf. "Optimization Methods". In: *Design of Modern Heuristics: Principles and Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 45–102. ISBN: 978-3-540-72962-4. URL: https://doi.org/10.1007/978-3-540-72962-4%E2%82%83.

[240] Hessam S Sarjoughian, Abdurrahman Alshareef, and Yonglin Lei. "Behavioral DEVS Metamodeling". In: *Winter Simulation Conference (WSC), 2015*. Huntington Beach, CA, USA, Dec. 6–9, 2015, pp. 2788–2799. ISBN: 978-1-4673-9743-8.

[241] August-Wilhelm Scheer. *Business Process Engineering: Reference Models for Industrial Enterprises*. Berlin; New York: Springer, 1998. ISBN: 978-3-662-03615-0.

[242] Andreas Schlegel, Johannes Stoldt, and Matthias Putz. "Erweiterte Integration Energetischer Betrachtungen in Der Materialflusssimulation". In: *15th ASIM Fachtagung Simulation in Produktion Und Logistik*. Ed. by Wilhelm Dangelmaier. Paderborn, Germany: ASIM, Oct. 9–11, 2013, pp. 187–196. ISBN: 978-3-942647-35-9. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.855.5049&rep=rep1&type=pdf.

[243] Artur Schmidt. *Variantenmanagement in Der Modellbildung Und Simulation Unter Verwendung Des SES/MB Frameworks*. ARGESIM, 2019. ISBN: 978-3-903347-30-4. DOI: 10.11128/fbs.30. URL: https://www.argesim.org/fbs30 (visited on June 8, 2020).

[244] Artur Schmidt and Thorsten Pawletta. "Hybride Modellierung Fertigungstechnischer Prozessketten Mit Energieaspekten in Einer Ereignisdiskreten Simulationsumgebung". In: *Proceedings of ASIM 2014 – 22. Symposium Simulationstechnik, Berlin, 03.-05.09.* 2014, pp. 109–116. URL: https://www.researchgate.net/profile/Thorsten_Pawletta/publication/266200212_Hybride_Modellierung_fertigungstechnischer_Prozessketten_mit_Energieaspekten_in_einer_ereignisorientierten_Simulationsumgebung/links/54296d3a0cf2e4ce940e6945.pdf (visited on Apr. 28, 2017).

[245] Robert Schmoll. *Co-Simulation und Solverkopplung: Analyse komplexer multiphysikalischer Systeme*. In collab. with Kassel University Press GmbH. Berichte des Instituts für Mechanik 3/2015. Kassel: Kassel University Press, 2015. 152 pp. ISBN: 978-3-86219-592-3.

[246] Michael Schneider, Andreas Stenger, and Dominik Goeke. "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations". In: *Transportation Science* 48.4 (Mar. 6, 2014), pp. 500–520. ISSN: 0041-1655. DOI: 10.1287/trsc.2013.0490. URL: https://pubsonline.informs.org/doi/abs/10.1287/trsc.2013.0490 (visited on Mar. 28, 2020).

[247] Malte Schönemann, Christopher Schmidt, Christoph Herrmann, and Sebastian Thiede. "Multi-Level Modeling and Simulation of Manufacturing Systems for Lightweight Automotive Components". In: *Procedia CIRP* 41 (2016), pp. 1049–1054. ISSN: 22128271. DOI: 10.1016/j.procir.2015.12.063. URL: https://linkinghub.elsevier.com/retrieve/pii/S2212827115011427 (visited on June 5, 2020).

[248] Günther Schuh, ed. *Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte*. 3., völlig neu bearb. Aufl. VDI-Buch. Berlin: Springer, 2006. 876 pp. ISBN: 978-3-540-40306-7.

[249] Günther Schuh and Volker Stich. *Produktionsplanung und -steuerung 2: Evolution der PPS*. Springer-Verlag, May 9, 2012. 434 pp. ISBN: 978-3-642-25427-7. Google Books: dhMeBAAAQBAJ.

[250] Fadi Shrouf, Joaquin Ordieres-Meré, Alvaro García-Sánchez, and Miguel Ortega-Mier. "Optimizing the Production Scheduling of a Single Machine to Minimize Total Energy Consumption Costs". In: *Journal of Cleaner Production* 67 (Mar. 15, 2014), pp. 197–207. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2013.12.024. URL: http://www.sciencedirect.com/science/article/pii/S0959652613008780 (visited on Feb. 22, 2020).

[251] Patrick Siarry. *Metaheuristics*. Springer, Dec. 24, 2016. 501 pp. ISBN: 978-3-319-45403-0. Google Books: 3dbJDQAAQBAJ.

[252] Wilfried Sihn, Thomas Sobottka, Bernhard Heinzl, and Felix Kamhuber. "Interdisciplinary Multi-Criteria Optimization Using Hybrid Simulation to Pursue Energy Efficiency through Production Planning". In: *CIRP Annals – Manufacturing Technology* 67.1 (June 2018), pp. 447–450. ISSN: 0007-8506. DOI: 10.1016/j.cirp.2018.04.059. URL: https://www.sciencedirect.com/science/article/pii/S0007850618300830 (visited on July 1, 2018).

[253] Carlos A. Silva, Thomas A. Runkler, João M. Sousa, and J. M. Sá da Costa. "Optimization of Logistic Processes in Supply-Chains Using Meta-Heuristics". In: *Progress in Artificial Intelligence*. Ed. by Fernando Moura Pires and Salvador Abreu. Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 2902. Lecture Notes in Computer Science (LNCS, Volume 2902). Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 9–23. ISBN: 978-3-540-20589-0. DOI: 10.

1007/978-3-540-24580-3_9. URL: http://link.springer.com/10.1007/978-3-540-24580-3_9 (visited on Mar. 6, 2020).

[254] Gregory A. Silver, Osama Al-Haj Hassan, and John A. Miller. "From Domain Ontologies to Modeling Ontologies to Executable Simulation Models". In: *Proceedings of the 2007 Winter Simulation Conference (WSC'07)*. 2007 Winter Simulation Conference. Dec. 2007, pp. 1108–1117. DOI: 10.1109/WSC.2007.4419710.

[255] Dan Simon. *Evolutionary Optimization Algorithms*. John Wiley & Sons, June 13, 2013. 776 pp. ISBN: 978-1-118-65950-2. Google Books: gwUwIEPqk30C.

[256] Peter Smolek. "Energy Modelling and Analysis of Industrial Production Facilities". PhD Thesis. Vienna, Austria: TU Wien, 2018. 90 pp. URL: http://katalog.ub.tuwien.ac.at/AC15243580 (visited on Feb. 19, 2020).

[257] Peter Smolek. "Objektorientierte Modellierung und dynamische Co-Simulation mit CATIA V6 am Beispiel von Kraftfahrzeugsystemen". Diploma Thesis. Wien: TU Wien, 2013. 122 pp. URL: http://www.ub.tuwien.ac.at/dipl/2013/AC11200075.pdf (visited on Oct. 8, 2016).

[258] Peter Smolek, Georgios Gourlis, Benjamin Mörzinger, Ines Leobner, and Karl Ponweiser. "Thermal Building Simulation of an Industrial Production Facility Using the Balanced Manufacturing Approach". In: *Proceedings of 12th Conference on Sustainable Development of Energy, Water and Environment Systems*. Dubrovnik, Croatia, Oct. 2017.

[259] Peter Smolek, Ines Leobner, Georgios Gourlis, Benjamin Mörzinger, Bernhard Heinzl, and Karl Ponweiser. "Hybrid Building Performance Simulation Models for Industrial Energy Efficiency Applications". In: *Journal of Sustainable Development of Energy, Water and Environment Systems* (2018).

[260] Peter Smolek, Ines Leobner, Bernhard Heinzl, Georgios Gourlis, and Karl Ponweiser. "A Method for Real-Time Aggregation of a Product Footprint During Manufacturing". In: *Journal of Sustainable Development of Energy, Water and Environment Systems* 4.4 (Dec. 2016), pp. 360–378. ISSN: 18489257. DOI: 10.13044/j.sdewes.2016.04.0028. URL: http://www.sdewes.org/jsdewes/pi2016.04.0028 (visited on Oct. 31, 2016).

[261] Thomas Sobottka. "Eine anwendungsorientierte simulationsbasierte Methode, unter Berücksichtigung von Energieeffizienz, in der optimierenden Planung von Produktion und Logistik". PhD Thesis. Vienna, Austria: TU Wien, 2017. 165 pp. URL: http://katalog.ub.tuwien.ac.at/AC14523544 (visited on Feb. 19, 2020).

[262] Thomas Sobottka, Felix Kamhuber, Mohammadali Faezirad, and Wilfried Sihn. "Potential for Machine Learning in Optimized Production Planning with Hybrid Simulation". In: *Procedia Manufacturing*. 25th International Conference on Production ResearchManufacturing Innovation: Cyber Physical ManufacturingAugust 9-14, 2019 | Chicago, Illinois (USA) 39 (2019), pp. 1844–1853. ISSN: 2351-9789. DOI:

10.1016/j.promfg.2020.01.254. URL: http://www.sciencedirect.com/science/article/pii/S2351978920303188 (visited on Mar. 9, 2020).

[263] Thomas Sobottka, Felix Kamhuber, Matthias Rößler, and Wilfried Sihn. "Hybrid Simulation-Based Optimization of Discrete Parts Manufacturing to Increase Energy Efficiency and Productivity". In: *15th Global Conference on Sustainable Manufacturing (GCSM 2017)*. Israel, Sept. 25–27, 2017.

[264] Thomas Sobottka, Felix Kamhuber, and Wilfried Sihn. "Increasing Energy Efficiency in Production Environments through an Optimized, Hybrid Simulation-Based Planning of Production and Its Periphery". In: *Proceedings of the 24th CIRP Conference on Life Cycle Engineering*. Kamakura, Japan, Mar. 8–10, 2017, pp. 440–445. URL: https://publik.tuwien.ac.at/files/publik_261080.pdf (visited on Dec. 20, 2017).

[265] Michael H. Spiegel. "Linking Simulation and Automation Infrastructure: A Study Based on the FMI and IEC 61499". Diploma Thesis. Wien, Austria: TU Wien, Mar. 2018.

[266] Michael H. Spiegel, Edmund Widl, Bernhard Heinzl, Wolfgang Kastner, and Nabil Akroud. "Model-Based Virtual Components in Event-Based Controls: Linking the FMI and IEC 61499". In: *Applied Sciences* 10.5 (Mar. 2020), p. 1611. DOI: 10.3390/app10051611. URL: https://www.mdpi.com/2076-3417/10/5/1611 (visited on Mar. 11, 2020).

[267] Statistik Austria. *Energiedaten Österreich 2016*. Vienna, Austria, 2017. URL: http://www.statistik.at/web_de/statistiken/energie_umwelt_innovation_mobilitaet/energie_und_umwelt/energie/energiebilanzen/index.html (visited on Feb. 21, 2020).

[268] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework*. 2nd ed. The Eclipse Series. Addison-Wesley Professional, 2008. 704 pp. ISBN: 978-0-321-33188-5.

[269] Perdita Stevens and R. J. Pooley. *Using UML: Software Engineering with Objects and Components*. Pearson Education, 2006. 276 pp. ISBN: 978-0-321-26967-6. Google Books: rDRE54yiwSUC.

[270] Michael Striebel. "Multirate - Introduction". Presentation. ARGESIM Seminar S300 - Multirate ODE/DAE Solver (TU Wien). May 31, 2011.

[271] Chris Swinerd and Ken R. McNaught. "Design Classes for Hybrid Simulations Involving Agent-Based and System Dynamics Models". In: *Simulation Modelling Practice and Theory* 25 (June 1, 2012), pp. 118–133. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2011.09.002. URL: http://www.sciencedirect.com/science/article/pii/S1569190X1100147X (visited on Jan. 29, 2020).

[272] J.R. Swisher, P.D. Hyden, S.H. Jacobson, and L.W. Schruben. "A Survey of Simulation Optimization Techniques and Procedures". In: *Proceedings of the 2000 Winter Simulation Conference*. 2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165). Vol. 1. Orlando, FL, USA: IEEE, Dec. 2000, pp. 119–128. DOI: 10.1109/WSC.2000.899706.

[273] Janos Sztipanovits and Gabor Karsai. "Model-Integrated Computing". In: *Computer* 30.4 (1997), pp. 110–111. URL: http://ieeexplore.ieee.org/abstract/document/585163/ (visited on Mar. 17, 2017).

[274] Antuela A Tako, Kathy Kotiadis, and Christos Vasilakis. "A Conceptual Modelling Framework for Stakeholder Participation in Simulation Studies". In: *Proceedings of the 2010 Operational Research Society Simulation Conference (SW10)*. Worcestershire, England: Operational Research Society, 2010, pp. 76–85.

[275] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. "Digital Twin in Industry: State-of-the-Art". In: *IEEE Transactions on Industrial Informatics* 15.4 (Apr. 2019), pp. 2405–2415. ISSN: 1551-3203, 1941-0050. DOI: 10.1109/TII.2018.2873186. URL: https://ieeexplore.ieee.org/document/8477101/ (visited on June 8, 2020).

[276] Jean-Philippe Tavella, Mathieu Caujolle, Charles Tan, Gilles Plessis, Mathieu Schumann, Stéphane Vialle, Cherifa Dad, Arnaud Cuccuru, and Sébastien Revol. "Toward an Hybrid Co-Simulation with the FMI-CS Standard". In: *hal-01301183* (Apr. 7, 2016). URL: https://hal-centralesupelec.archives-ouvertes.fr/hal-01301183 (visited on Feb. 12, 2020).

[277] The MathWorks. *MATLAB Documentation*. Version 9.7 (R2019b). Natick, Massachusetts: The MathWorks Inc., 2019. URL: http://www.mathworks.com/help/matlab/index.html.

[278] The MathWorks. *MATLAB Version 9.7 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[279] The MathWorks. *Simscape User's Guide*. Sept. 2019. URL: https://de.mathworks.com/help/pdf_doc/physmod/simscape/simscape_ug.pdf.

[280] Sebastian Thiede. *Energy Efficiency in Manufacturing Systems*. Sustainable Production, Life Cycle Engineering and Management. Berlin, Heidelberg: Springer, 2012. ISBN: 978-3-642-25913-5. DOI: 10.1007/978-3-642-25914-2. URL: http://link.springer.com/10.1007/978-3-642-25914-2 (visited on Jan. 29, 2020).

[281] Sebastian Thiede, Malte Schönemann, Denis Kurle, and Christoph Herrmann. "Multi-Level Simulation in Manufacturing Companies: The Water-Energy Nexus Case". In: *Journal of Cleaner Production* 139 (Dec. 15, 2016), pp. 1118–1127. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2016.08.144. URL: http://www.sciencedirect.com/science/article/pii/S0959652616313087 (visited on Mar. 8, 2020).

[282] Sebastian Thiede, Yingying Seow, Jon Andersson, and Björn Johansson. "Environmental Aspects in Manufacturing System Modelling and Simulation—State of the Art and Research Perspectives". In: *CIRP Journal of Manufacturing Science and Technology* 6.1 (Jan. 2013), pp. 78–87. ISSN: 17555817. DOI: 10.1016/j.cirpj.2012.10.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1755581712000806 (visited on June 5, 2020).

[283] Michael Tiller. *Introduction to Physical Modeling with Modelica*. Springer Science & Business Media, Dec. 6, 2012. 348 pp. ISBN: 978-1-4615-1561-6. Google Books: 7vcGCAAAQBAJ.

[284] Stavros Tripakis. "Bridging the Semantic Gap between Heterogeneous Modeling Formalisms and FMI". In: *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS). July 2015, pp. 60–69. DOI: 10.1109/SAMOS.2015.7363660.

[285] U.S. Department of Energy. *EnergyPlus Version 9.2.0 Documentation: Engineering Reference*. Version 9.2.0. U.S. Department of Energy, Sept. 27, 2019. URL: https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v9.2.0/EngineeringReference.pdf.

[286] U.S. Department of Energy. *EnergyPlus Version 9.2.0 Documentation: Getting Started*. Version 9.2.0. U.S. Department of Energy, Sept. 27, 2019. URL: https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v9.2.0/GettingStarted.pdf (visited on Feb. 18, 2020).

[287] Thomas H. -J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. "The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0". In: *Procedia CIRP*. The 24th CIRP Conference on Life Cycle Engineering 61 (2017), pp. 335–340. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.11.152. URL: http://www.sciencedirect.com/science/article/pii/S2212827116313129 (visited on June 8, 2020).

[288] Thomas H. -J. Uhlemann, Christoph Schock, Christian Lehmann, Stefan Freiberger, and Rolf Steinhilper. "The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems". In: *Procedia Manufacturing*. 7th Conference on Learning Factories, CLF 2017 9 (2017), pp. 113–120. ISSN: 2351-9789. DOI: 10.1016/j.promfg.2017.04.043. URL: http://www.sciencedirect.com/science/article/pii/S2351978917301610 (visited on June 8, 2020).

[289] R. Vahrenkamp. *Produktionsmanagement*. Oldenbourg, 2008. ISBN: 978-3-486-58784-5.

[290] Peter J. M. van Laarhoven and Emile H. L. Aarts. "Simulated Annealing". In: *Simulated Annealing: Theory and Applications*. Ed. by Peter J. M. van Laarhoven and Emile H. L. Aarts. Mathematics and Its Applications. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-94-015-7744-1. DOI: 10.1007/978-94-

015-7744-1_2. URL: https://doi.org/10.1007/978-94-015-7744-1_2 (visited on Mar. 18, 2020).

[291] Lars Völker. *Untersuchung des Kommunikationsintervalls bei der gekoppelten Simulation*. Karlsruher Schriftenreihe Fahrzeugsystemtechnik Bd. 6. Karlsruhe: KIT Scientific Publ, 2011. 174 pp. ISBN: 978-3-86644-611-3.

[292] Markus Völter, Thomas Stahl, Jorn Bettin, Arno Haase, and Simon Helsen. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, June 26, 2013. 495 pp. ISBN: 978-1-118-72576-4. Google Books: 9ww_D9fAKncC.

[293] T.F. Wallace. *Sales and Operations Planning: The How-to Handbook, 2nd Ed.* T. F. Wallace & Company, 2004. ISBN: 978-0-9674884-4-8.

[294] Zhaojie Wang, Feng Gao, Qiaozhu Zhai, Xiaohong Guan, Kun Liu, and Dianmin Zhou. "An Integrated Optimization Model for Generation and Batch Production Load Scheduling in Energy Intensive Enterprise". In: *Proceedings of the 2012 IEEE Power and Energy Society General Meeting*. 2012 IEEE Power & Energy Society General Meeting. New Energy Horizons - Opportunities and Challenges. San Diego, CA: IEEE, July 2012, pp. 1–8. ISBN: 978-1-4673-2729-9. DOI: 10.1109/PESGM.2012.6345296. URL: http://ieeexplore.ieee.org/document/6345296/ (visited on June 7, 2020).

[295] Ezra Wari and Weihang Zhu. "A Survey on Metaheuristics for Optimization in Food Manufacturing Industry". In: *Applied Soft Computing* 46 (Sept. 2016), pp. 328–343. ISSN: 15684946. DOI: 10.1016/j.asoc.2016.04.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S156849461630182X (visited on Mar. 8, 2020).

[296] Jos B. Warmer and Anneke G. Kleppe. *The Object Constraint Language: Getting Your Models Ready for MDA*. Addison-Wesley Professional, 2003. 242 pp. ISBN: 978-0-321-17936-4. Google Books: gu_6L_hVIfEC.

[297] John W. Webb and Ronald A. Reis. *Programmable Logic Controllers: Principles and Applications*. Prentice Hall, 1999. 472 pp. ISBN: 978-0-13-679408-0. Google Books: z9dSAAAAMAAJ.

[298] Peter Wegner. "Concepts and Paradigms of Object-Oriented Programming". In: *ACM SIGPLAN OOPS Messenger* 1.1 (Aug. 1, 1990), pp. 7–87. ISSN: 1055-6400. DOI: 10.1145/382192.383004. URL: https://doi.org/10.1145/382192.383004 (visited on May 11, 2020).

[299] Nils Weinert. *Vorgehensweise für Planung und Betrieb energieeffizienter Produktionssysteme*. Ed. by Günther Seliger. Berichte aus dem Produktionstechnischen Zentrum Berlin. Stuttgart: Fraunhofer-Verl, 2010. 143 pp. ISBN: 978-3-8396-0173-0.

[300] Nils Weinert, Stylianos Chiotellis, and Günther Seliger. “Methodology for Planning and Operating Energy-Efficient Production Systems”. In: *CIRP Annals* 60.1 (2011), pp. 41–44. ISSN: 00078506. DOI: 10.1016/j.cirp.2011.03.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0007850611000163 (visited on June 6, 2020).

[301] Sigrid Wenzel, Tim Peter, Johannes Stoldt, Andreas Schlegel, Tobias Uhlig, and János Jósvai. “Considering Energy in the Simulation of Manufacturing Systems”. In: *2018 Winter Simulation Conference (WSC)*. 2018 Winter Simulation Conference (WSC). Dec. 2018, pp. 3275–3286. DOI: 10.1109/WSC.2018.8632238.

[302] Engelbert Westkämper. *Einführung in die Organisation der Produktion*. Springer-Lehrbuch. Berlin/Heidelberg: Springer-Verlag, 2006. ISBN: 978-3-540-26039-4. DOI: 10.1007/3-540-30764-8. URL: http://dx.doi.org/10.1007/3-540-30764-8 (visited on Feb. 23, 2020).

[303] Michael Wetter. “Co-Simulation of Building Energy and Control Systems with the Building Controls Virtual Test Bed”. In: *Journal of Building Performance Simulation* 4.3 (Sept. 2011), pp. 185–203. ISSN: 1940-1493, 1940-1507. DOI: 10.1080/19401493.2010.518631. URL: http://www.tandfonline.com/doi/abs/10.1080/19401493.2010.518631 (visited on Sept. 27, 2016).

[304] Michael Wetter and Thierry Nouidui. *Building Controls Virtual Test Bed - User Manual Version 1.6.0*. Version 1.6.0. Berkeley, CA: Lawrence Berkeley National Laboratory, Apr. 20, 2016. URL: http://simulationresearch.lbl.gov/bcvtb/releases/1.6.0/doc/manual/bcvtb-manual.pdf.

[305] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. ISBN: 978-3-662-43838-1. DOI: 10.1007/978-3-662-43839-8. URL: http://link.springer.com/10.1007/978-3-662-43839-8 (visited on June 8, 2020).

[306] Daniel Wolff, Dennis Kulus, and Stefan Dreher. “Simulating Energy Consumption in Automotive Industries”. In: *Use Cases of Discrete Event Simulation: Appliance and Research*. Ed. by Steffen Bangsow. Berlin, Heidelberg: Springer, 2012, pp. 59–86. ISBN: 978-3-642-28777-0. DOI: 10.1007/978-3-642-28777-0_4. URL: https://doi.org/10.1007/978-3-642-28777-0_4 (visited on June 7, 2020).

[307] A. Wayne Wymore. *Model-Based Systems Engineering*. CRC Press, May 4, 2018. 659 pp. ISBN: 978-1-351-43108-8. Google Books: 00taDwAAQBAJ.

[308] Lei Xiujuan and Shi Zhongke. “Overview of Multi-Objective Optimization Methods”. In: *Journal of Systems Engineering and Electronics* 15.2 (June 2004), pp. 142–146. ISSN: 1004-4132.

[309] Betul Yagmahan and Mehmet Mutlu Yenisey. "A Multi-Objective Ant Colony System Algorithm for Flow Shop Scheduling Problem". In: *Expert Systems with Applications* 37.2 (Mar. 1, 2010), pp. 1361–1368. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.06.105. URL: http://www.sciencedirect.com/science/article/pii/S0957417409006605 (visited on Mar. 31, 2020).

[310] Virginia Yannibelli and Analía Amandi. "Hybridizing a Multi-Objective Simulated Annealing Algorithm with a Multi-Objective Evolutionary Algorithm to Solve a Multi-Objective Project Scheduling Problem". In: *Expert Systems with Applications* 40.7 (June 2013), pp. 2421–2434. ISSN: 09574174. DOI: 10.1016/j.eswa.2012.10.058. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417412011827 (visited on Mar. 18, 2020).

[311] Mesut Yavuz, Elif AKçali, and Süleyman Tüfekçi. "Optimizing Production Smoothing Decisions via Batch Selection for Mixed-Model Just-in-Time Manufacturing Systems with Arbitrary Setup and Processing Times". In: *International Journal of Production Research* 44.15 (Aug. 1, 2006), pp. 3061–3081. ISSN: 0020-7543. DOI: 10.1080/00207540500478454. URL: https://doi.org/10.1080/00207540500478454 (visited on Mar. 18, 2020).

[312] M. Yazdani, M. Amiri, and M. Zandieh. "Flexible Job-Shop Scheduling with Parallel Variable Neighborhood Search Algorithm". In: *Expert Systems with Applications* 37.1 (Jan. 1, 2010), pp. 678–687. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.06.007. URL: http://www.sciencedirect.com/science/article/pii/S0957417409005685 (visited on Mar. 8, 2020).

[313] Faruk Yilmaz, Umut Durak, Koray Taylan, and Halit Oğuztüzün. "Adapting Functional Mockup Units for HLA-Compliant Distributed Simulation". In: The 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden. Mar. 10, 2014, pp. 247–257. DOI: 10.3384/ecp14096247. URL: http://www.ep.liu.se/ecp/article.asp?issue=096%26article=25 (visited on Feb. 12, 2020).

[314] Lei Yonglin, Wang Weiping, Li Qun, and Zhu Yifan. "A Transformation Model from DEVS to SMP2 Based on MDA". In: *Simulation Modelling Practice and Theory* 17.10 (Nov. 1, 2009), pp. 1690–1709. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2009.08.003. URL: http://www.sciencedirect.com/science/article/pii/S1569190X09001117 (visited on May 4, 2020).

[315] Lei Yonglin, Wang Weiping, Li Qun, and Zhu Yifan. "A Transformation Model from DEVS to SMP2 Based on MDA". In: *Simulation Modelling Practice and Theory* 17.10 (2009), pp. 1690–1709. DOI: 10.1016/j.simpat.2009.08.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S1569190X09001117 (visited on July 2, 2018).

[316] Serge Zacher. *Automatisierungstechnik kompakt*. Vieweg+Teubner Verlag, May 30, 2000. 524 pp. ISBN: 978-3-528-03897-7. Google Books: H1YtE9lW_foC.

[317] Günther Zäpfel, Roland Braune, and Michael Bögl. *Metaheuristic Search Concepts*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-11342-0. DOI: 10.1007/978-3-642-11343-7. URL: https://doi.org/10.1007/978-3-642-11343-7 (visited on Mar. 18, 2020).

[318] Bernard P. Zeigler. "Embedding DEV&DESS in DEVS". In: *DEVS Integrative Modeling & Simulation Symposium*. Vol. 7. 2006, p. 18.

[319] Bernard P. Zeigler, Alexandre Muzy, and Ernesto Kofman. *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Third edition. London San Diego Cambridge, MA Oxford, UK: Academic Press, 2019. 667 pp. ISBN: 978-0-12-813370-5. Google Books: VBhpDwAAQBAJ.

[320] Bernard P. Zeigler, Herbert Prähofer, and Tag Gon Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. USA: Academic Press, 2000. 536 pp. ISBN: 978-0-12-778455-7.

[321] F. Zezulka, P. Marcon, Z. Bradac, J. Arm, T. Benesl, and I. Vesely. "Communication Systems for Industry 4.0 and the IIoT". In: *IFAC-PapersOnLine*. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018 51.6 (Jan. 1, 2018), pp. 150–155. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.07.145. URL: http://www.sciencedirect.com/science/article/pii/S2405896318308899 (visited on Feb. 29, 2020).

[322] Li Zheng and H. Nakagawa. "OPC (OLE for Process Control) Specification and Its Developments". In: *Proceedings of the 41st SICE Annual Conference (SICE 2002)*. Proceedings of the 41st SICE Annual Conference. SICE 2002. Vol. 2. Osaka, Japan, Aug. 2002, pp. 917–920. DOI: 10.1109/SICE.2002.1195286.

[323] W.H.M. Zijm. "Towards Intelligent Manufacturing Planning and Control Systems". In: *OR-Spektrum* 22.3 (Aug. 1, 2000), pp. 313–345. ISSN: 1436-6304. DOI: 10.1007/s002919900032. URL: https://doi.org/10.1007/s002919900032 (visited on Mar. 4, 2020).