Agent-based Discrete-Event Simulation Environment for Electric Power Distribution System Analysis

A dissertation approved for the academic degree of

Doctor of Engineering (Dr.-Ing.)

at the

Faculty of Electrical Engineering
and Information Technology

of

TU Dortmund University

by

Johannes Hiry, M.Sc.
from Fürweiler

October 2021

Supervisor:	Univ.-Prof. Dr.-Ing. habil. Christian Rehtanz
Co-Advisor:	Univ.-Prof. Dr. Sebastian Lehnhoff
Day of the oral examination:	22.12.2021

Abstract

The ongoing transformation of the entire power system with a simultaneous convergence of different, previously separated energy sectors poses new challenges, especially for the energy transport infrastructure at the distribution level. Due to its important role for society as a whole, careful planning and stable, secure operation of this infrastructure are essential. Accordingly, the introduction of necessary innovative operational concepts or the exploration of new planning approaches in the context of the energy transition cannot take place in the real system.

One possibility to evaluate new, innovative methods without endangering the real system is simulation. However, the complexity of the electrical infrastructure at the distribution level, the large number of heterogeneous technical systems and the influence of individual, human-centric behavior pose a number of challenges for existing approaches.

In the context of the present work, an agent-based modeling approach is combined with a discrete-event simulation approach to form the agent-based discrete-event simulation model SIMONA. Based on experiences gained with existing preliminary work, an existing agent-based model is revised, remodeled and implemented from scratch, discrete-event simulation logic is introduced, and further necessary adjustments are made. The resulting model enables large-scale simulations of the electrical distribution grid while taking into account individual behavior of connected grid assets, innovative grid operation concepts and flexible system participant behavior. As a result, the developed simulation model can generate extensive, detailed time series usable for grid planning, operation, and analysis purposes.

Kurzfassung

Die fortlaufende Transformation des gesamten Energiesystems bei gleichzeitigem Zusammenwachsen unterschiedlicher, bisher getrennter Energiesektoren stellt insbesondere die Energietransportinfrastruktur auf der Verteilnetzebene vor neue Herausforderungen. Aufgrund ihrer wichtigen Bedeutung für das gesamtgesellschaftliche Leben ist eine sorgfältige Planung und ein stabiler, sicherer Betrieb dieser Infrastruktur unerlässlich. Die Einführung notwendiger innovativer betrieblicher Konzepte oder die Erprobung neuer Plannungsansätze im Rahmen der Energiewende kann entsprechend nicht im realen System erfolgen.

Simulationen stellen eine Möglichkeit dar, neue, innovative Methoden ohne eine Gefährdung des realen Systems zu erproben. Die Komplexität der elektrischen Infrastruktur auf der Verteilnetzebene, die große Anzahl heterogener technische Anlagen sowie die Beeinflussung durch individuelles, menschliches Verhalten stellt entsprechende bestehende Ansätze jedoch vor eine Reihe von Herausforderungen.

Im Rahmen der vorliegenden Arbeit wird ein agentenbasierter Modellierungsansatz mit einem ereignisdiskreten Simulationsansatz zum agentenbasierten, ereignisdiskreten Simulationsmodell SIMONA kombiniert. Basierend auf Erfahrungen mit Vorarbeiten wird ein bestehendes agentenbasiertes Modell von Grund auf neu modelliert und implementiert, eine ereignisdiskrete Simulationslogik eingeführt und weitere notwendige Anpassungen vorgenommen. Das resultierende Modell ermöglicht großskalige Simulationen des elektrischen Verteilnetzes bei gleichzeitiger Berücksichtigung individuellen Verhaltens der einzelnen Netzanschlussnehmer, innovativer Netzbetriebskonzepte und flexiblen Anlagenverhaltens. Im Ergebnis kann das entwickelte Simulationsmodell umfangreiche, detaillierte Zeitreihen generieren, die für Netzplanung, -betrieb und -analyse genutzt werden können.

Acknowledgment

This thesis results from research activities conducted during my time as a research associate at the Institute of Energy Systems, Energy Efficiency and Energy Economics (ie^3) at TU Dortmund University.

I want to express my deepest gratitude to my supervisor Univ.-Prof. Dr.-Ing. habil. Christian Rehtanz, who made this work possible in the first place. While working on this thesis, he was always available for me as a constructive-critical discussion partner in his inimitable friendly manner. I thank him for the extensive trust he has placed in me and the resulting freedom he has given me in working on research projects. As a scientific employee and research group leader at his institute, the assigned tasks have contributed significantly to my personal development.

I want to thank Univ.-Prof. Dr. Sebastian Lehnhoff for his willingness to take over the co-advisory. The interdisciplinary nature of this thesis and its nexus between electrical engineering and computer science, particularly benefited from his helpful comments.

Thanks to my colleagues at ie^3 for the enjoyable work atmosphere, great moments at scientific conferences, and the personal friendships that arose from this time. In addition, I thank all students for their support in different research projects, either as student assistants or in the context of bachelor's or master's theses.

I also sincerely thank my colleagues at Lawrence Berkeley National Laboratory for the amazing experiences during my research stay at Berkeley and the countless opportunities to think outside the box.

From among my colleagues, I want to express my special gratitude to Dr.-Ing. Chris Kittl. Not only for the successful cooperation in several research projects, but especially for the evolved friendship, his honest character, his urge to question things critically, but also his willingness to embrace new challenges.

I further thank all friends and colleagues who worked through the manuscript of this thesis and made valuable contributions to its improvement.

A special thanks goes to my parents for their everlasting support and who sacrificially cared for my education up to the doctorate degree. A big thank you also goes to the rest of my family for their constant encouragement and support.

"We absolutely must leave room for doubt or there is no progress and there is no learning. There is no learning without having to pose a question. And a question requires doubt."
– *Richard P. Feynman*, 1964

Contents

1 Introduction

For more than three decades, the energy system has faced a fundamental paradigm shift towards a more sustainable, socio-ecological power system. With the social movements against nuclear power and the demand for less polluting and more environmental-friendly electricity generation, the first phase of the energy system transition began in the late 1980s. Starting with a slowly growing substitution of fossil fuels by renewable distributed energy resources (DER), the adoption of the German Renwable Energy Sources Act (EEG) in 2000 has accelerated the utilization of DER in Germany. Additionally, the EEG and the German energy transition have become a blueprint for the transformation of the electricity generation sector in the world. [1] With the motivation to reduce global greenhouse gas emissions, almost all other industrial nations implemented common targets and incentives to prevent or at least reduce global warming. [2] As a consequence of these decisions, the electric energy system, particularly power distribution systems, are becoming more complex. [3] In order to guarantee a reliable energy supply, the volatile feed-in of DER, their high level of weather-dependent feed-in simultaneity, as well as the inversion of the previously top-down power flow in the distribution grid to a bottom-up power flow, requires new measures. Consequently, innovative methods for planning and operating electrical grids, such as the *BDEW traffic light concept* [4], [5] or *peak capping* [6], were developed to achieve this.

The rising number of electric vehicles, new sector-coupling assets, e.g., power-to-gas assets to convert and store electricity generated from DER and increased usage of power-to-heat appliances, marked the beginning of the second phase of the energy system transformation (Figure 1.1). In addition, advances in Information and Communication Technology (ICT) enable an increased development of *smart grid* and *smart market* applications. A shift of power system boundaries and a trend towards an increasing interconnection of formerly mostly separated energy sectors electricity, gas, heat and mobility is observable. Coupling these sectors is complex, involves many stakeholders and gains more and more relevance.

Although the full impact of this development is not yet foreseeable today, it already raises questions. For example, how can a large number of technical systems be integrated into existing technical and regulatory structures? The impact of public, semi-public or private charging stations on the electrical grid will most likely differ from the impact of heat pumps or battery systems. Individual, human-centered usage behavior, e.g., mobility patterns or the future development of mobility in general, will have an increasing impact on distribution systems.

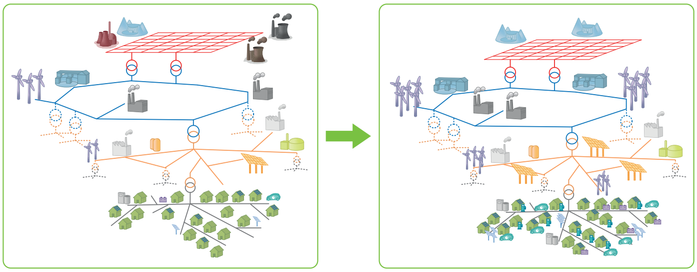


Figure 1.1: Schematic overview on the energy system transformation at the distribution system

Additionally, it is necessary to find answers to questions about the operation of system participants. Which effects will a market-driven or a grid-oriented operation have on the distribution system? What is the relationship between incentive mechanisms for charging behavior and their impact on the distribution grid? Which control options are needed by system operators to intervene in a minimally invasive but efficient manner? Which technical prerequisites are needed for such situations regarding ICT as well as manageability of a large number of loads and feed-ins?

With the primary task to operate a secure and stable distribution system, answering these and other questions is of central importance for Distribution Grid Operators (DSOs). Moreover, they are particularly facing additional follow-up questions. For instance, how can the flexible behavior of different system participants be quantified for consideration in grid planning and operation? How do costs for ICT and flexibility exploitation compare to conventional grid reinforcement measures? Which risks result from a reduced grid reserve and how can these risks be reduced to a minimum? However, to adequately answer these questions and find the best solution, different options and their impact must first be investigated.

New, innovative models of active distribution systems, capturing the systems' critical properties, are required to enable these investigations. Among others, these properties comprise individual human and technical behavior, many heterogeneous, decentralized system participants and their time-dependent operation. The interdisciplinary and sector-crossing nature of active distribution systems requires explicit modeling of large numbers of system participants and not only aggregated equivalent models. Only that way it is possible to investigate potentially emergent behavior of the overall system.

Multi-Agent Systems (MASs) allow for such a detailed representation of complex systems[1] and their entities using the principle of decomposition. Their application in the context of Agent-based Models (ABMs) and Agent-based Simulations (ABSs) allow the generation time series and thus the consideration of temporal dependencies. Therefore, the combination of agent theory and time series generation seems to be a promising approach for modeling and simulating active distribution systems and to answer research questions related to their analysis, planning, and operation.

This thesis is about the representation of the power distribution system as part of the emerging interconnected overall system using agent-based modeling and discrete-event simulation techniques to generate detailed system time-series.

Employing Agent-based Simulation

According to [8], ABMs have proven to be viable in both academia and practice for the study of complex, highly heterogeneous systems. Their flexible structure enables interdisciplinary use in the form of MAS in the field and as ABS for simulations. Their ability to capture all relevant features of a complex system by decomposing it into small sub-problems makes it very attractive to create comprehensible models. Figure 1.2 provides a schematic overview of the general ABM components.

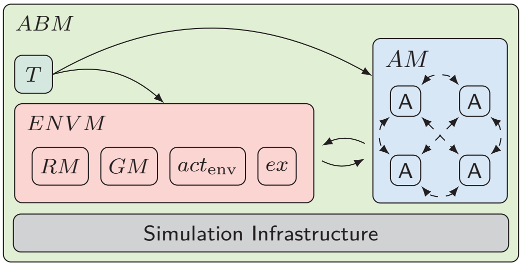


Figure 1.2: Schematic overview of the general components of an ABM

More formally, an ABM is an invariant structure

$$ABM = \langle T, ENVM, AM \rangle \tag{1.1}$$

[1]Complex systems in the context of this thesis are systems with the properties presented in [7].

where T is the representation of time within the ABM, $ENVM$ is the model for the environment of the MAS and AM is the actual model of the MAS. The environment model is further decomposable into a model of all resources RM, the description of global properties of the environment GM, a global action function $act_{\text{env}}(\cdot)$ that defines the actions the environment is capable of, e.g., move forward in time, and an execution function $ex(\cdot)$, responsible for the effect of the actions of the agents.

$$ENVM = \langle RM, GM, act_{\text{env}}, ex \rangle \tag{1.2}$$

When applied for modeling complex systems, the structure of agent-based approaches is an advantage and a disadvantage. Its flexibility allows its application in various domains, but it also requires a careful model formulation that can be challenging. In particular, the precise definition of communication and behavior protocols for the agents' interaction, as well as finding a good balance between required model detail and acceptable abstraction, is crucial. Otherwise, the results might not be detailed enough to be interpreted correctly or lead to pseudo accuracy and large computational effort. However, properly modeled and implemented, MASs and ABMs are useful methods to analyze complex systems. [8]

The simulation model developed in [9] and [10] demonstrates that an agent-based approach is suitable for distribution grid modeling. However, with its objective to integrate DER, smart grid and smart market mechanisms in the planning process, the presented model only addresses the demands of the first phase of the power system transformation. First of all, the model's existing implementation uses outdated libraries that prevent its utilization for large-scale simulations, making it unusable for distribution system simulations with thousands of system participants. Furthermore, no special attention has been paid to sectors beyond the distribution grid at the time of model development. Nevertheless, progressing sector coupling tendencies require a more flexible simulation approach considering emerging interdependencies, e.g., through co-simulations. Finally, the pure planning perspective is no longer sufficient anymore. Instead, a model is needed that supports grid planning, simulates operation and enables a holistic system analysis.

In view of the above, the main research question of this thesis is as follows:

How to model and implement an efficient and flexible distribution power system simulation with a focus on the electricity grid that is capable of serving as an analysis and development testbed to generate time series for modern planning, operation and analysis purposes?

Outline and Contributions

The presented thesis consists of six chapters. In the following, the content of each chapter is briefly outlined, and its contributions are highlighted.

Chapter 2 – Fundamentals and Trends in the Power Distribution System

Chapter 2 introduces the reader to the fundamentals of electric power systems with the primary focus on distribution grid planning and operation. The chapter starts with an analysis of the changing role of the DSO caused by the challenges arising from the latest developments in the energy system transformation process. The impact of these challenges on distribution grid analysis, planning and operation are outlined and it is shown how time series may be a supportive approach for DSOs. Afterward, the mathematical fundamentals concerning electricity grid modeling and power flow calculations are presented and different methods to solve complex power flow equations are outlined. The chapter closes with a review of the state of the art of different commercial and freely available tools to calculate steady-state power flows, system participant usage profiles, as well as co-simulation frameworks allowing for sector-coupled and large-scale power distribution system simulations.

Chapter 3 – Modeling and Simulation Fundamentals

Chapter 3 provides the methodical basics that form the foundation for the subsequent model development. It contains a short introduction to the fundamentals of modeling and simulation in general. The differences between the terms *modeling* and *simulation* are explained in order to simplify the understanding of the following chapters. Additionally, the chapter briefly introduces the reader to the different types of models and simulations and defines several common terms. A particular focus is put on the notion of time within simulations. In this context, a short introduction to discrete-event modeling and simulation approaches is given, including the required fundamentals used in this thesis. In the last part of this chapter, differences and similarities between MAS, ABM and ABS as well as their specific attributes are discussed. Furthermore, different ABS implementation concepts are presented.

Chapter 4 – Agent-based Discrete-Event Electric Power System Model

Chapter 4 presents the main contributions of this thesis: an agent-based discrete-event model of the power system at the distribution grid level shown in Figure 1.3. The chapter starts with a specification and requirement description of the model in Section 4.1, followed by a conceptual overview of the newly developed overall model in Section 4.2 and a classification of agents in the context of the developed model in Section 4.3. Afterward, selected parts of the overall system developed in the context of this thesis are described in detail.

Section 4.4 introduces the reader to the model of the developed discrete-event scheduling mechanism, which is responsible for the temporal synchronization and administration of all agents and the environment model during a simulation run.

Since a particular focus is on the detailed representation of the electric power system, Section 4.5 presents the model of a Distributed Backward-Forward Sweep Algorithm (DBFS) algorithm to solve power flow equations in the context of an agent-based simulation. It starts with a description of the corresponding mathematical preliminary considerations and the general functioning of the algorithm. Afterward, the algorithm transformation from a centralized to a decentralized solution and its modeling in an agent-based context, including the agent's interaction and communication protocols, are presented in detail. The result of this section is a newly developed power flow method, which can execute concurrent large-scale multi-voltage-level simulations supporting two winding and three winding transformers at grid coupling points and a variety of grid topologies.

Section 4.6 then describes the general functionality of agents that represent system participants connected to the electricity grid using an illustrative example of a specific model of a *Photovoltaic (PV) agent* (*PvAgent*). The description puts a particular focus on a newly developed, physical PV unit model and the verification and validation of its power feed-in calculations. This model is required as the previously existing model presented in [9] has issues capturing steep angles of solar radiation.

The chapter closes with an evaluation of the developed model with regard to specifications and requirements defined in Section 4.1.

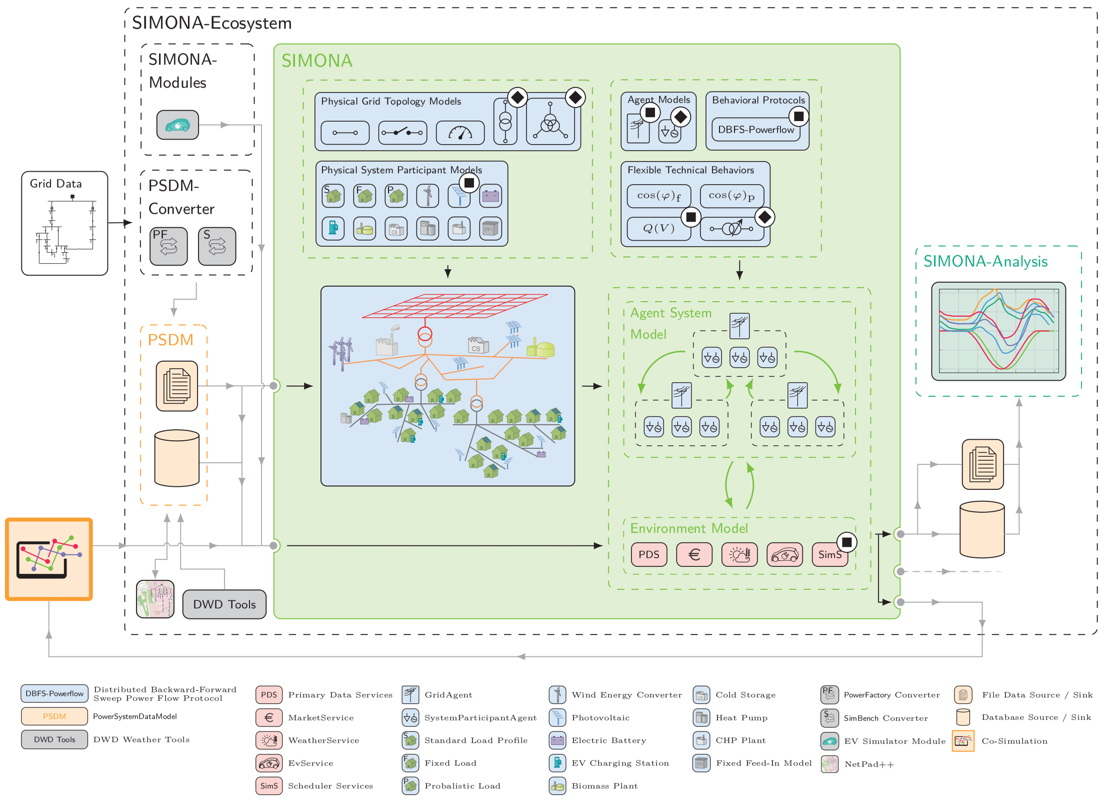


Figure 1.3: Overview of SIMONA and related modules of the SIMONA-Ecosystem

In summary, the contributions of Chapter 4 are

- a discrete-event agent-based simulation model for large scale power system simulations at the distribution level called SIMONA;
- a formal description of simulated agents in the context of power system simulation used in SIMONA;
- a formal concept of a scheduling mechanism for agent-based discrete-event simulations and time synchronization;
- a formal grid decomposition and partitioning approach for multi-voltage-level grids supporting two and three winding transformers at grid coupling points;
- an agent-based model of a backward-forward sweep algorithm including the corresponding communication and interaction protocols;
- a highly detailed physical model of PV units used in the *PvAgent*;

Being part of extensive research activities within different research projects, SIMONA and the SIMONA-Ecosystem has been designed, developed and implemented together with my research partner Chris Kittl. While this thesis focuses on the particular parts of Simona mentioned above, [11] presents further details on the overall system, in particular on system participants and their behaviors, two- and three-winding transformer models and their validation, and comparison with already established models. Contributions to SIMONA presented in the context of both dissertations are marked in Figure 1.3 with ▣ for this thesis or ◈ for [11], respectively.

While the overall system has been developed jointly as part of the research activities of both authors, the respective dissertation publications present only selected aspects of the system in detail. Therefore, in addition to the dissertations, the publications [P8], [P7], [P5] present additional components and functions of SIMONA. Appendix D.1 outlines some details about the SIMONA-Ecosystem and provides references to the OpenSource (OS) code of several components. Current work in progress prepares the publication of supplementary research articles of, until now, unpublished components. In addition, so far not publicly available parts of the code of SIMONA and the SIMONA-Ecosystem will be made available to the public soon under a OS license.

Chapter 5 – Application Example

In Chapter 5, the developed agent-based simulation system is applied for power system simulations in the context of two application examples. In the first application example, a large number of SimBench [12] benchmark grids are simulated and the results are compared with the provided benchmark power flow reference data. This comparison allows an extensive verification and validation of the simulation and demonstrates the proper operation of SIMONA. In the second, more practice-oriented application case, the developed model is applied to investigate the impact of a connection request of a PV power plant park with two expansion stages. Within the simulation, externally provided time series are combined with internal PV model calculations to execute multi-voltage level simulations to generate grid utilization time series. Moreover, to demonstrate the possibility to consider flexibility in SIMONA simulations, the impact of a $Q(V)$ voltage setpoint control to the PV park grid connection request is investigated.

In summary, the contributions of Chapter 5 are

- the verification and validation of the proper functionality of SIMONA;
- a demonstration of the model's capabilities based on a real-world use case including the simulation of flexibility;

Chapter 6 summarizes the main findings, critically reviews the developed model and gives an outlook for possible future work.

2 Fundamentals and Trends in Power Distribution Systems

The detailed and accurate modeling of the electric power distribution system represents an elementary building block for its analysis, planning and operation. The increasing coupling of different sectors with the electric power system requires a more detailed consideration in the modeling and simulation process of the overall system. If done carefully, it enables Distribution Grid Operators (DSOs) to plan and operate their grids in a secure, efficient and cost-effective way in the context of the transformation from passive to active distribution grids. Concerning the specific requirements of such simulations, steady-state power flow calculations and time series allow adequate consideration of flexible varying processes in the grid, including impacts resulting from sector coupling. In addition to grid time series, asset behavior and usage time series are also required in an appropriate level of detail, including electric system participants and assets from coupled sectors. However, tools and approaches currently available do not or only partly provide these capabilities.

In this chapter, an introduction to the current challenges at the electric power distribution grid level followed by an outline of the impacts on DSOs is given. Furthermore, a mathematical model of the electricity grid and different methods for power flow calculations for a specific grid operation points is presented. The chapter closes with an overview of the current state of the art in the field of electric power system modeling and simulation, including different commercial and OpenSource (OS) steady-state power flow solving tools, different system participant behavioral models as well as co-simulation frameworks available for large scale sector-coupled power system simulations.

2.1 Changing Role of Distribution Grid Operators

With the ongoing energy transformation and the corresponding changes at the power distribution grid level, in particular, DSOs are confronted with a series of new challenges. These challenges not only need to be addressed but actively incorporated by DSOs. Consequently, DSOs are more and more transforming from a predominantly passive to a more active role with increased system responsibility.

This section outlines different selected challenges for the electric distribution grid, followed by a brief description of the resulting implications for DSOs.

2.1.1 Challenges for the Electric Distribution Grid

In an increasingly interconnected digital world, the electric power system is a central pillar of modern societies. Electricity plays a role in almost all areas of daily life - and the trend is rising. While in the past, due to centralized electricity production by a few large power plants, the transmission grid played an important role, with the ongoing energy transformation, the significance of the distribution grid more and more increases as well.

Focusing on Germany, the roots of the energy transition can be traced back to the 1980s, where people started to demand more environmental-friendly and sustainable electricity generation. Twenty years later, in 2000, the introduction of the EEG [1] accelerated the transformation from centralized to decentralized energy generation and since then, the share of distributed energy resourcess (DERs) has strongly increased. In contrast to large-scale power plants connected at the transmission grid, DERs are mainly connected to the distribution grid. This situation causes a change in the previously unidirectional power flow from the transmission to distribution level to a now bi-directional power flow between both levels.[2] In parallel to the ongoing integration of DER the next phase of the energy system transformation has already started - the decarbonization of the mobility sector and the coupling of different energy sectors forming a new coupled energy system at the distribution grid level. The number of Electric Vehicles (EVs) and corresponding charging infrastructure started growing already for a few years. Additionally, new assets like electric batteries or sector-coupling-storages [13] are increasingly available. With a few exceptions, all these assets are connected at the distribution grid level - either on the electricity, heat or gas side.

As a consequence of these developments, the importance of the distribution system and electricity grid in particular more and more increases. Additionally, the inherent structural changes from a predominantly passive to an active distribution grid also referred to as *smart grids*, introduce several new challenges with regard to its planning and operation. Specifying them is not trivial as many of them overlap in individual aspects or are mutually dependent - another indication of the complexity of the resulting system. The following overview can therefore only represent a small part of the overall spectrum and should be seen as selected exemplary changes and resulting challenges:

[2] Transmission grids are considered to cover voltage levels with a nominal voltage of 220 up to 400 kV while distribution grids are considered to cover voltage levels with a nominal voltage of 0.4 up to 110 kV.

Changed Supply Task New assets like DERs, EVs, battery electric storages, heat pumps and other sector-coupling resources as well as prosumer households drastically change the traditional supply task. The trend to decarbonization increased the need for electricity and, without proper energy management, may subsequently result in increased peak load.

Increased System Complexity The rising number of assets also increases the complexity of the distribution system structure and its operation. While in the first phase of the energy transition, only electricity-related assets like Wind Energy Converter (WEC) and Photovoltaic (PV) units were connected to the grid at a large scale, with the new sector coupling units, a new level of complexity due to newly induced interdependencies is reached. Although different options consider these developments in distribution grid operation like energy management or load curtailment, their large-scale application for thousands of grid assets connected at the low-voltage is quite challenging and increases the grid complexity even further.

Increased Volatility With the higher amount of different assets, the power generation and consumption, in general, becomes more volatile. DER feed-in and, to some extend, utilization of electrically operated heat appliances mainly depends on weather conditions, but the electricity consumption of EVs through charging stations depends on human mobility behavior. This situation can even become more complex if currently discussed flexibility markets come into play. The resulting grid utilization can only be estimated and forecasted to a certain extend, and the remaining uncertainty requires proper consideration.

Statefulness and Temporal Coupling With the connection of EVs, electricity storages and sector-coupling-storages to the distribution grid, a new dependency is introduced. In the past, almost all grid-connected assets have been stateless, meaning that their functionality is only influenced by their current state. New assets, however, may have a time- and operation-dependent state which influences their operation capabilities. Examples of such assets are electric batteries or heat storage, whose State of Charge (SoC) depends on feed-in-consumption cycles during the last few hours or days. Again, flexibility markets and their time-dependent impact on the grid may introduce another variable. In conclusion, statefulness can either result from external events, e.g., regulation by law or market situations, or internally based on technical requirements or boundaries.

Lack of Grid Transparency Regarding their operation, transmission and distribution systems differ not only on their voltage level. While transmission systems have been planned and operated precisely and computer-aided for several decades, distribution systems, comprising primary and secondary technologies and all other connected systems, are far simpler structured. Their passive nature did not require any measurement devices, sensors or Information and Communication Technology (ICT) so far, in particular not at the low and medium-voltage level. Hence, while becoming more and more important, the predominant part of the distribution grid lacks a sufficient level of grid transparency and controllability. In consequence, local overloads or instabilities may not be detected or resolved.

2.1.2 Impact on Distribution System Operators

With the ongoing transformation from a passive to an active distribution system, the role of the distribution grid operator is also evolving from a predominantly passive to a more intervening, active one. Consequently, these developments require the advancement of traditional distribution grid planning and operation concepts. An essential task in this context is the increasing consideration of operational aspects in the grid planning process. However, the relationship between grid planning and operation is proving to be increasingly ambivalent. This ambivalence stems from the fact that the planning process becomes more cost-efficient by considering operational aspects such as curtailment and flexibility in general, but at the same time, the actual operational grid reserve is decreased by this consideration. In turn, a decreasing grid reserve results in an increased risk concerning a secure and stable grid operation. Furthermore, to assure that the grid operates within its boundaries, it is crucial to verify that operational actions considered in the planning process are available in the respective grid and vice versa. Depending on their risk aversion, DSOs will have to find their balance between the consideration of operational concepts in planning and the associated increase in planning accuracy regarding costs and grid utilization as well as the resulting increased risk in grid operation due to a lower operational grid reserve (Figure 2.1).

Distribution System Planning and Operation

There are already various approaches in science and practice, which try to address the described challenges within distribution grid planning and operation. While

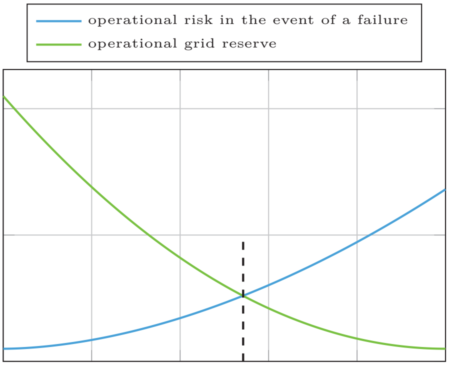


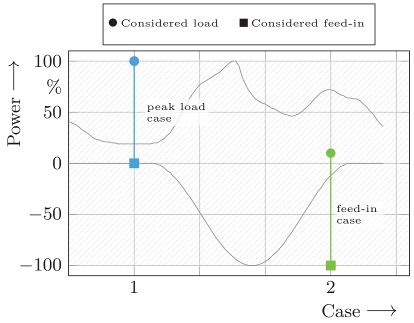
Figure 2.1: Schematic tradeoff between operational reserve and grid planning efficiency

some have already found their way into practice, others are part of ongoing research projects. Figure 2.2 provides a schematic overview of selected approaches.

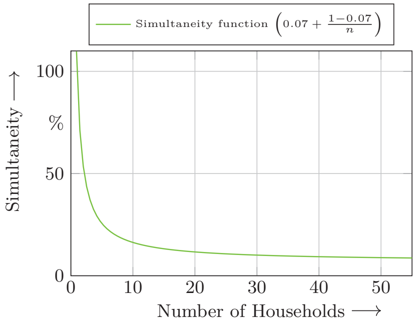
For instance, the conventional *fit and forget* planning approach (Figure 2.2a) using two extreme scenarios, maximum load combined with minimum feed-in and maximum feed-in combined with minimum load, is not suitable anymore. [15]

As a result, the planning process has been evolved in several steps. With the introduction of *simultaneity factors*, exemplarily depicted as simultaneity function in Figure 2.2b, the attempt to incorporate volatile new loads or feed-ins in the conventional planning process was made. These factors are determined once using a series of Monte Carlo simulations and can be integrated comparably easily into the existing, traditional distribution grid planning process. [16] However, while being relatively simple in its application, this approach only captures the volatility of new assets but fails in the exploitation of operational flexibility in the planning process.

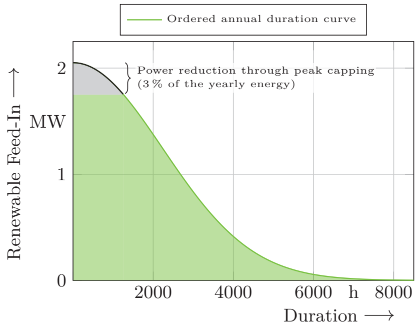
Hence, a second approach, *peak capping*, was made available by law in Germany a few years ago with the main target to consider operational curtailment in the planning process. Figure 2.2c exemplarily demonstrates the fixed percentage peak capping approach visually. The proposed concept allows for the reduction of annually produced energy of PV and WEC units up to 3 %. As legislators do not specify the exact method, different approaches to calculate and consider the annually reducible energy amount may apply. [14] To integrate peak capping in the planning



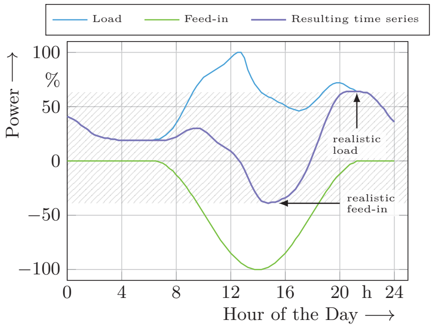
(a) Fit and forget planning with extreme scenarios



(b) Exemplary simultaneity function



(c) Exemplary fixed percentage peak capping using an ordered annual duration curve based on [14]



(d) Time series based planning considering a realistic supply task

Figure 2.2: Schematic overview of different grid planning approaches

process, annual time series of the respective WEC or PV units are required. In contrast to simultaneity factors, which require time series only once for their determination, time series are always mandatory within the application of peak capping in the planning process. May it be for the determination of general, nationwide peak-capping reduction factors or the detailed consideration of specific WEC or PV units in a specific grid. Peak capping thus represents, for the first time, a significant extension of the conventional grid planning process and breaks with the sole consideration of two extreme scenarios.

Besides applying time series, the general advancement of distribution grid planning approaches is an ongoing field of research. Accordingly, there are several comprehensive overviews on the current state of the art, e.g., [17]–[19]. An evaluation of the different methodologies, their advantages and drawbacks is presented in [9] and [14]. Both evaluations have in common that they mention time series as one possible way to capture attributes of modern, active distribution grids, as shown in Figure 2.2d to make them available in the distribution grid planning process. One reason for this may be that they can be used flexibly, either in the context of a pure time series-based distribution grid planning, e.g., as an independent methodology, as a preliminary analysis and parametrization step or as input data for another method. In [10], the impact of reactive power management together with the traffic light concept on grid expansion demand of a voltage level coupled medium and low-voltage is investigated using a time series based power flow over one year. [20] evaluates the consideration of the use of active power curtailment and reactive power control in distribution system planning. Also, for their application in a preliminary analysis step or input data, several examples exist. For example, in studies presented in [21] and [22], which are estimating the German distribution grid expansion needs, time series are used to model DER curtailment. [23] and [24] make use of smart meter measures in order to determine grid utilization and derive planning requirements.

Similar developments are also observable in the operation of distribution grids. First of all, any operational flexibility considered in planning must also be implementable in operation. Additionally, as already indicated, general grid operation and system services are also becoming significantly more demanding and require new approaches. In particular, the stronger interconnection on the horizontal distribution system level and the shift of generation capacity from the transmission level to the distribution level resulting in bi-directional power flows also come with a shift of responsibilities. While in the past Transmission Grid Operators (TSOs) have been the main active part, the increased responsibility of the DSOs results in an increased need for communication between system-level operators in daily operation planning and during disturbances or failures. More than 800 independent DSOs in Germany alone makes defining and implementing these communication requirements quite challenging. Time series are one of the main building blocks which allow for a shift from so far passive DSOs to active, intervening one.

For example, the integration of DER and prosumers leads to an additional need for operational flexibility in order to ensure that the sum of active power injection and consumption can be balanced and that the grid can be operated within a specified frequency band. However, these requirements necessitate further development of operational management processes and horizontal as well as vertical data exchange between plant operators, DSOs and TSOs. With the amendment to the Grid Expansion Acceleration Act (NABEG 2.0) [25] on 01.10.2021, the regulatory framework and the grid operation process will be adapted accordingly. Figure 2.3 illustrates this process schematically for electricity grid operators. For the sake of simplification, additional, non-grid-related roles involved in this process, e.g., direct marketers or balancing group managers, are neglected there.

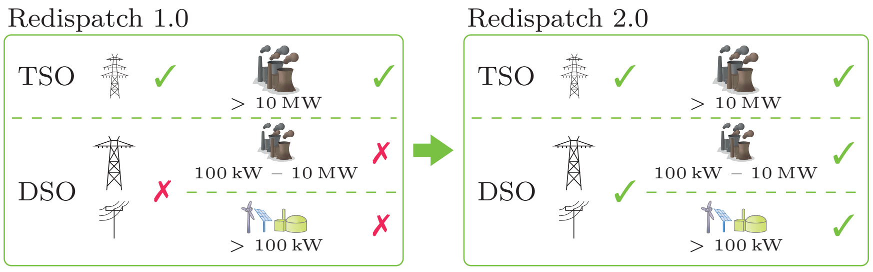


Figure 2.3: Horizontal integration of grid operation responsibilities in the context of the Grid Expansion Acceleration Act (NABEG 2.0)

The so-called *Redispatch 2.0* will enable the decentralization for operational planning and control processes in order to avoid grid congestion across all grid levels with the goal to accomplish a more efficient, economical and resilient grid operation. One of its main ingredients is the transformation of the, up to now separated, feed-in management into a holistically organized process. The number of plants involved in this process will increase significantly with the additional consideration of smaller conventional power plants with a feed-in capacity of 100 kW to of 10 MW and DER with a feed-in capacity greater than 100 kW. As a result, an increased need for communication and coordination effort between DSOs and TSOs is expected.

In order to enable distribution grid operators to participate in this process, a certain degree of grid transparency is required. Grid transparency means that several measures already applied at the transmission grid level and sometimes on the high-voltage level at the distribution grid now need to be extended up to the low-voltage level. These measures include the implementation of more measurement infrastructure and sensors to collect data on the current state of the grid, as well as, e.g., the set-up of central control centers or decentralized control units. Central control centers may continuously monitor all voltage levels in the distribution grid and are able to perform operational day-ahead planning. Additionally, decentralized control units may be used for autonomous control and protection tasks in the field. These units would only report to central control rooms if a failure or significant issue occurs to avoid information overload for the people responsible there. While many TSOs and a few DSOs already base their operational planning processes on load, feed-in and weather forecast data, most of the DSOs do not yet. However, as demonstrated, the significance of forecasts at the distribution grid level is increasing. [26]

Additionally, existing standard load profiles may also be expected to be updated in order to account for the new heterogeneity of devices connected to the distribution grid. They are also neither able nor designed to capture the behavior of single households but are only valid for a sufficiently large number of them. A possible way to further increase grid transparency may be the installation of smart meters. Their data may additionally be used to develop new standard load profiles. Though their large-scale implementation in the field will not be completed until 2028. [26]

In order to ensure continuous grid transparency as well as to derive new standard load profiles, time-sequential data is required. This data may reflect the grid's current state based on measurement data and is also required for day-ahead planning purposes. As it is currently not or only partly available from the field, simulation models that can calculate and estimate the grid state considering several short-term future scenarios as part of the day-ahead planning process are required.

Distribution System Modelling and Simulation

The fact that the electrical energy system at the distribution grid level is a central infrastructure for society requires a careful approach to its transformation to ensure a secure and safe supply of energy now and in the future. Consequently, to plan energy transformation pathways and understand their impact on power distribution systems, detailed holistic models are required before new approaches are implementable in the field. As these simulations need to be as accurate as possible to capture all crucial aspects of the modeled system, single extreme scenario-based simulations are unsuitable. Instead, particularly to correctly represent time-dependent operating assets, they need to simulate multiple points in time whereby corresponding result data must subsequently be available as time series. These time series could be grid utilization, asset feed-in or consumption, weather data or any other data of interest captured in a timely ordered sequence. Depending on the actual use case, their resolution might vary between seconds to hours. [26], [27]

Depending on the availability of measurement infrastructure in a specific grid, simulated time series can be further augmented using real measurement values. In that case, different application cases of measured data and simulated time series are imaginable. First, historical measurements could be used to develop or parametrize simulation models, subsequently allowing for what-if scenario analysis. Secondly, it would be possible to build hybrid simulations from real-time field measured data combined with artificially simulated data of areas without installed measurement infrastructure to estimate the grid state. In contrast to the conventional state estimation, this approach would also allow hybrid what-if analysis considering measured real-world data and artificially simulated changes in other assets behavior.

From the perspective of the power distribution system, in particular, three application fields for such simulations can be identified:

Distribution System Planning In the planning process of active distribution grids, the determination of single grid operation points is not suitable anymore. Instead, it is required to consider operational flexibility, market interactions, new sector-coupling assets and resulting sectoral-interdependencies as input parameters of the planning process. This consideration requires detailed models that can be used to simulate the grid utilization and generate grid, asset feed-in and consumption time series. To a certain extent, it is additionally required to generate time series of assets located in other, coupled sectors. Using co-simulations with domain-specialized simulation tools and associated single domain models allows for such a detailed consideration of other sectors and their interdependencies for time series generation. [28]

Distribution System Operation In the actual distribution system operation process, again, a variety of simulations is required to represent and analyze the current state of the grid. First of all, as the planning process determines the actual physical boundaries of the grid operation, whatever has been considered in the planning process needs to be known during grid operation. If flexibility, e.g., from curtailment operation, demand-side management or similar, has been assumed to be available during the grid planning needs to be accessible during its operation. As already stated, the reduced operational grid reserve due to advanced planning approaches needs careful consideration during day-ahead planning, grid state estimation and additional operational measures. Furthermore, in order to use advanced methods of grid operation, e.g., neural state estimation [29], detailed simulated time series need to be available as well either as input data or for further analyses. If in the future *smart markets* are implemented at the distribution grid level, their impact also needs to be taken into account accordingly using simulations. [26]

Distribution System Research Research in the power distribution system field is crucial, and the energy system is too important to execute field experiments without adequate detailed a priori testing. Hence, whenever new distribution system planning or operation approaches are developed, they need to be tested extensively using simulation approaches. Particularly extreme cases, whose probability of occurrence in reality is relatively low, can thus be checked for newly developed algorithms and their robustness can be ensured.

2.1.3 Summary

As the energy transition advances, electrical distribution grids and their operators face various new challenges. In contrast to previous developments in the energy transition context, other horizontal sectors such as gas, heat, and mobility become more important. Consequently, the planning and operation of the power distribution system is becoming more complex and requires extensive further developments. Simulation models play a significant role here. The execution of simulations allows the analysis and evaluation of different scenarios and thus represent an essential basis for planning and operation decision-making. Due to the time-dependent statefulness of new grid resources and the interdependencies resulting from sector coupling, in contrast to the past considering a few extreme scenarios is no longer sufficient. Instead, it is necessary to perform time-sequential simulations to generate various asset and grid time series.

2.2 Complex Power Flow Calculations

Regardless of the concrete object of investigation, two components are of central importance for the simulation of the electricity grid. First of all, a precise representation of the grid and all connected system participants is required. Furthermore, depending on the available grid state information, methods and algorithms for solving power flow equations are necessary to determine the stationary power flow for a particular operating point. The following explanations are therefore divided into two parts. First, an electrical grid in a power supply system is formally described as a mathematical model of power flow equations. Secondly, different standard solution methods from the literature are presented based on these equations.

2.2.1 Mathematical Description of the Electric Grid State

The determination of the quasi-stationary grid state is a common approach in grid analysis, planning and operation. The underlying grid model is assumed to be in a quasi-stationary state, which means that no dynamic transient processes or control interventions occur anymore. This assumption simplifies the model description and subsequently the solution of the model's equations.

Based on the assumption of a grid in a quasi-stationary state, the grid operating point is defined of the grid power flows, the consumption or feed-in of connected system participants and the corresponding node voltages and currents. This grid operating point can be used to determine the grid's utilization which depends on the different physical limits of nodes, lines and transformers as well as the specific design of the grid topology, e.g., meshing degree or voltage level. A steady-state three-phase grid in symmetric operation can further be simplified to a single-phase system. [30] This simplification makes the resulting model describable using algebraic equations. The subsequent non-linear grid state equations are mainly based on [31] including some modifications from [14].

In its simplest form, an abstract model of the electrical grid can be understood as an undirected, weighted graph. Here, the graph's vertices correspond to the electrical grid nodes and the edges correspond to all connecting grid elements such as lines and transformers. The following sections first describe the general, formal relationships between the physical properties of the grid nodes and edges. Based on this, the linking of vertices and edges is described formally, and a classification of the grid nodes into different subsets is carried out. As the main focus of this thesis is the analysis of the electric grid, the term node, if not indicated otherwise, is from now on used synonymously for a grid topology vertex. The term edge is replaced with the more common electrical term branch for all connecting grid assets, such as lines or transformers. All referenced physical quantities in the following are written as lower case and are given in p.u.. For a comprehensive overview of the mathematical nomenclature, please refer to the List of Symbols.

In order to calculate the power flows over the individual grid elements, it is first necessary to determine the power that is feed-in or consumed by the connected system participants at the respective grid nodes. Let $\mathbf{N}_g$ be the set of all nodes of an electric grid g, such that $\mathbf{N}_g = \{n_1, n_2, n_3, ..., n_m\}$ and $m = |\mathbf{N}_g|$. Let i be a single node, such that $i \in \mathbf{N}_g$. The apparent power at this single node i can be described as the difference between the complex power feed-in $\underline{s}_{\mathrm{G},i}$ and the complex load $\underline{s}_{\mathrm{L},i}$ at the respective node depicted in Equation 2.1.

$$\underline{s}_i = \underline{s}_{\mathrm{G},i} - \underline{s}_{\mathrm{L},i} = p_i + \mathrm{j}q_i \tag{2.1}$$

Additionally, the apparent power of a single node i can be calculated using the complex node voltage $\underline{v}_i$ and the conjugate complex node current $\underline{i}_i^*$, which flows into, or out from the system participants connected at i as shown in Equation 2.2.

$$\underline{s}_i = \underline{v}_i \cdot \underline{i}_i^* \tag{2.2}$$

Accordingly, Equation 2.3 describes the apparent power of the whole grid g with all nodes $i \in \mathbf{N}_g$.

$$\underline{\vec{s}}_g = \underline{\vec{v}}_g \circ \underline{\vec{i}}_g^* \tag{2.3}$$

Following Kirchhoff's current law, the sum of all currents flowing into node i must be equal to the sum of all currents flowing out of that node. Considering this, the current at node i can be formulated by the sum of the current flowing into node i from all other nodes $j \neq i$ (Equation 2.4). [32], [33]

$$\underline{i}_i = \sum_{\forall j \in \mathbf{N}, j \neq i} \underline{i}_{ij} \tag{2.4}$$

Applying Equation 2.4 to Equation 2.2 leads to Equation 2.5 which allows the calculation of the apparent nodal power based on the voltage at node i and the sum of the conjugate complex currents flowing into node i.

$$\underline{s}_i = \underline{v}_i \sum_{\forall j \in \mathbf{N}, j \neq i} \underline{i}_{ij}^* \tag{2.5}$$

In addition to the equations describing the nodes of a grid, its branches must also be formally described to enable a grid state determination. Assuming symmetrical operation, grid branches can be modeled either by using a T- or a π-equivalent circuit diagram, while the latter is depicted in Figure 2.4.

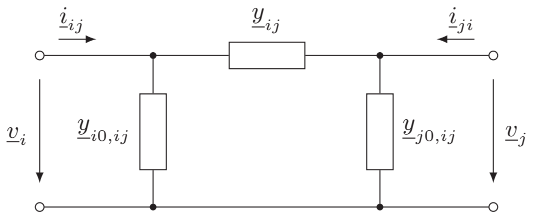


Figure 2.4: Generic grid branch π-equivalent circuit diagram

To describe the physical attributes of a branch, a similar notation as used for the node descriptions can be used. Let $\mathbf{B}_g$ be the set of all branches of an electric grid, such that $\mathbf{B}_g = \{b_1, b_2, b_3, ..., b_l\}$ and $l = |\mathbf{B}_g|$. The branch element $b_{ij} \in \mathbf{B}_g$ then connects two distinct nodes i and j such that $i, j \in \mathbf{N}_g \wedge i \neq j$. There, $\underline{y}_{i0,ij}$ then denotes the admittance to ground potential at node i, $\underline{y}_{j0,ij}$ denotes the admittance to ground potential at node j and $\underline{y}_{ij}$ denotes the admittance between node i and node j of the branch b_{ij}.

The complex apparent power $\underline{s}_{ij}$ which occurs on a branch b_{ij}, also called power flow between node i and j or j and i, respectively, can be determined using Equation 2.6 and Equation 2.7.

$$\underline{s}_{ij}^* = p_{ij} - \mathrm{j}q_{ij} = \underline{v}_i^* \cdot i_{ij} \tag{2.6}$$

$$\underline{s}_{ji}^* = p_{ji} - \mathrm{j}q_{ji} = \underline{v}_j^* \cdot i_{ji} \tag{2.7}$$

The needed currents between node i and j and vice versa can be calcaluted using Equation 2.8 and Equation 2.9.

$$\underline{i}_{ij} = (\underline{v}_i - \underline{v}_j) \cdot \underline{y}_{ij} + \underline{v}_i \cdot \underline{y}_{i0,ij} \tag{2.8}$$

$$\underline{i}_{ji} = (\underline{v}_j - \underline{v}_i) \cdot \underline{y}_{ij} + \underline{v}_j \cdot \underline{y}_{j0,ij} \tag{2.9}$$

Using Equation 2.8 and Equation 2.9 accordingly to replace the current on the branches, Equation 2.6 and Equation 2.7 become:

$$\underline{s}_{ij}^{*} = \underline{v}_{i}^{*} \cdot (\underline{v}_{i} - \underline{v}_{j}) \cdot \underline{y}_{ij} + v_{i}^{2} \cdot \underline{y}_{i0,ij} \tag{2.10}$$

$$\underline{s}_{ji}^{*} = \underline{v}_{j}^{*} \cdot (\underline{v}_{j} - \underline{v}_{i}) \cdot \underline{y}_{ij} + v_{j}^{2} \cdot \underline{y}_{j0,ij} \tag{2.11}$$

Given the apparent power flows, the complex transmission losses of the considered branch can be determined applying Equation 2.12 carefully taking into account the correct sign. The total losses $\underline{s}_{\text{loss},g}$ of a grid g, given in Equation 2.13, then correspond to the sum of the individual losses $\underline{s}_{\text{loss},b_l}$ of all branches in $\mathbf{B}_g$.

$$\underline{s}_{\text{loss},b_{ij}} = \underline{s}_{\text{loss},ij} = \underline{s}_{ij} + \underline{s}_{ji} \tag{2.12}$$

$$\underline{s}_{\text{loss},g} = \sum_{\forall b \in \mathbf{B}_g} \underline{s}_{\text{loss},b_l} \tag{2.13}$$

For the sake of completeness, it should be mentioned that by applying Equation 2.8, Equation 2.4 can be transformed into Equation 2.14.

$$\underline{i}_{i} = \sum_{\forall j \in \mathbf{N}_g, j \neq i} (\underline{v}_{i} - \underline{v}_{j}) \cdot \underline{y}_{ij} + \underline{v}_{i} \cdot \underline{y}_{i0,ij} \tag{2.14}$$

Alternatively to Equation 2.3, the whole grid g can be described based on the relationship between node currents, node voltages and the grid element admittances. For this purpose, the node admittance matrix $\left[\underline{Y}_{\text{N},g}\right]^{m \times m}$ is used, which links the vectors of node voltages $\underline{\vec{v}}_g$ and currents $\underline{\vec{i}}_g$ as matrix-vector multiplication as given in Equation 2.15.

$$\underline{\vec{i}}_g = \left[\underline{Y}_{\text{N},g}\right] \cdot \underline{\vec{v}}_g \tag{2.15}$$

The vectors $\underline{\vec{i}}_g = (\underline{i}_1, \underline{i}_2, \underline{i}_3, ..., \underline{i}_m)^{\mathsf{T}}$ and $\underline{\vec{v}}_g = (\underline{v}_1, \underline{v}_2, \underline{v}_3, ..., \underline{v}_m)^{\mathsf{T}}$ thereby reflect the complex node currents and node voltages for all grid nodes.

The node admittance matrix can be constructed according to Equation 2.16 and Equation 2.17. Its construction rules distinguish between main diagonal elements with indices $i = j$ and other elements with indices $i \neq j$. The admittance value of the main diagonal $\underline{y}_{\mathrm{N},ii}$ is the sum of all admittances connected to the node i (Equation 2.16). For all other elements $\underline{y}_{\mathrm{N},ij} = \underline{y}_{\mathrm{N},ji}$ either the negative admittance between the nodes i and j or zero, in case of no connection, is used. [32]

$$\underline{y}_{\mathrm{N},ii} = \sum_{\forall j \in \mathbf{N}_g, j \neq i} \left(\underline{y}_{ij} + \underline{y}_{i0,ij} \right) \tag{2.16}$$

$$\underline{y}_{\mathrm{N},ij} = \underline{y}_{\mathrm{N},ji} = -\underline{y}_{ij}, \quad \forall j \in \mathbf{N}_g, j \neq i \tag{2.17}$$

The relationship between node currents, voltages and admittances obtained from the node admittance matrix can now be used to replace the node currents in Equation 2.5. The apparent power $\underline{s}_i$ can be calculated accordingly using the voltage magnitude and the admittances at node i as presented in Equation 2.18.

$$\underline{s}_i = \underline{v}_i \sum_{\forall j \in \mathbf{N}_g, j \neq i} \underline{y}_{ij}^* \cdot \underline{v}_j^* \tag{2.18}$$

Correspondingly, Equation 2.3 can be formulated for the apparent power $\vec{\underline{s}}_g$ of the whole grid as given in Equation 2.19.

$$\vec{\underline{s}}_g = \vec{\underline{v}}_g \circ \left(\left[\underline{Y}_{\mathrm{N},g} \right] \cdot \vec{\underline{v}}_g \right)^* \tag{2.19}$$

With the previous descriptions of the physical attributes of electrical grid nodes, branches and their relationship, a system of non-linear algebraic equations is given. From the non-linear system of equations six partly unknown state variables result for each node i of the grid. These state variables are the node voltage magnitude v_i and the corresponding node voltage angle δ_i, active power $p_{\mathrm{L},i}$ and reactive power $q_{\mathrm{L},i}$ consumption resulting from connected loads at node i as well as active power $p_{\mathrm{G},i}$ and reactive power $q_{\mathrm{G},i}$ resulting from connected feed-in system participants. In order to solve the grid equations, four of the six possible state variables need to be known for each node. Following [31], the individual nodes can be divided into three categories as shown in Table 2.1.

Nodes of type *pq* represent the majority in a distribution grid. Usually, either loads or generators without voltage regulation capabilities are connected to this type of node. At pq-nodes, active and reactive power are known for the load and feed-in, and the node voltage magnitude and angle are unknown.
Nodes with voltage regulation capability are classified as *pv*-nodes. The known

Table 2.1: Node classification according to [31]

Node type	**Known variables**	**Unknown variables**
pq	$p_{\mathrm{L},i}$, $q_{\mathrm{L},i}$, $p_{\mathrm{G},i}$, $q_{\mathrm{G},i}$	v_i, δ_i
pv	v_i, $p_{\mathrm{L},i}$, $q_{\mathrm{L},i}$, $p_{\mathrm{G},i}$	$q_{\mathrm{G},i}$, δ_i
reference/slack	v_i, δ_i, $p_{\mathrm{L},i}$, $q_{\mathrm{L},i}$	$p_{\mathrm{G},i}$, $q_{\mathrm{G},i}$

variables at this node type are the active and reactive power of the loads, generators' active power and the node voltage magnitude. The reactive generator power and the node angle are unknown.

The so-called *slack* or *reference* node type comprises all nodes connected to an external, static grid. This grid can either be a superior grid or a large power plant whose properties allow the assumption to be almost static. Slack nodes available to a grid are used to balance the grid's power, including its losses. By definition, the voltage at this node type corresponds to the defined nominal voltage of the grid level in which the node is located with a voltage angle of zero degrees. Most of the time, distribution grids only have one slack node, but multiple ones are also possible, e.g., in the case of multiple connections to a superior grid.

Based on the classification in Table 2.1, the description of the set $\mathbf{N}_g$ of all nodes in a given grid g can be specified accordingly, such that

$$\mathbf{N}_g = \mathbf{N}_{\mathrm{pq}} + \mathbf{N}_{\mathrm{pv}} + \mathbf{N}_{\mathrm{s}} \tag{2.20}$$

where $\mathbf{N}_{\mathrm{pq}}$ is the set of all pq-nodes, $\mathbf{N}_{\mathrm{pv}}$ is the set of all pv-nodes and $\mathbf{N}_{\mathrm{s}}$ is the set of all slack nodes.

The state of all nodes, branches, and an entire grid can be defined and described through the preceding explanations. In order to use the resulting description to determine a specific grid state, it is necessary to solve the non-linear system of equations. For this purpose, different solution methods are available, which are described briefly in the following.

2.2.2 Mathematical Methods to Solve Power Flow Equations

The previous chapter presents one possible mathematical model of the electrical grid, usable determine the grid state through power flow calculations and generate grid time series. In order to perform these calculations, several different methods, including long-established numerical methods such as the Gauss-Seidel method [34], the Newton-Raphson Method (NR-Method) [35] or the Fast-Decoupled Load Flow method [36] are available. Additional other approaches are, for example, backward-forward sweep methods [37]–[39], the Holomorphic Embedding Load Flow [40] or the use of Artificial Neural Networks [41]. Within this thesis, the NR-Method, as well as a basic understanding of the general functionality of backward-forward sweep methods, are of particular importance. Therefore, the following section shortly introduces both approaches with regard to the aspects relevant to this thesis. For further explanations on power flow solving methods, please refer to relevant literature such as [31], [34], [42], [43].

Newton-Raphson method

The NR-Method is a generic, iterative approximation algorithm used to solve non-linear equation systems numerically. Starting with an initial guess $x^{(0)}$ as a root of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, it successively provides a better approximation of the function's final solution. This is achieved by relaxing the original function using a first-order Taylor expansion as linerization. [31], [34] For an n-dimensional vector this can can be done using Equation 2.21. $\left[J_f\left(\vec{x}^{\,(k)}\right)\right]$ is the matrix of partial derivates or Jacobian matrix of the function $f(\cdot)$ at the point $\vec{x}^{\,(k)}$ and k denotes the number of the algorithm iteration.

$$f\left(\vec{x}\right) \approx f\left(\vec{x}^{\,(k)}\right) + \left[J_f\left(\vec{x}^{\,(k)}\right)\right] \cdot \left(\vec{x} - \vec{x}^{\,(k)}\right) \tag{2.21}$$

The next approximation, $\vec{x}^{\,(k+1)}$, can be obtained by inserting it into Equation 2.21 and subsequently the x-intercept determination of the resulting function such that $f\left(\vec{x}^{\,(k+1)}\right) \overset{!}{=} \vec{0}$ holds. Hence, with

$$f\left(\vec{x}^{\,(k)}\right) + \left[J_f\left(\vec{x}^{\,(k)}\right)\right] \cdot \left(\vec{x}^{\,(k+1)} - \vec{x}^{\,(k)}\right) \overset{!}{=} \vec{0} \tag{2.22}$$

which can be solved to

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \left[J_f\left(\vec{x}^{(k)}\right)\right]^{-1} \cdot f\left(\vec{x}^{(k)}\right) \tag{2.23}$$

the recursion rule for several iterations can be defined.

Equation 2.23 is then recursively applied in each iteration of the NR-Method until the differences between the current and the previous iteration $\left|\Delta\vec{x}^{(k)}\right| = \left|\vec{x}^{(k)} - \vec{x}^{(k+1)}\right|$ is equal or smaller than a predefined convergence threshold ε. As the Jacobian matrix needs to be inverted in each iteration, this process may be computationally expensive. [31], [34]

The application of the NR-Method on an unidimensional function $f(x) = x^2$ is illustrated in Figure 2.5. Starting with the initial guess $x^{(0)}$ the function $f(x)$ is linearized using Equation 2.21. The application of Equation 2.23 then leads to the first improved solution $x^{(1)}$ which can be used to calculate $\left|\Delta x^{(0)}\right| = \left|x^{(0)} - x^{(1)}\right|$ in order to execute the convergence check. This process is repeated until the predefined convergence criterion ε is fulfilled. [31], [34]

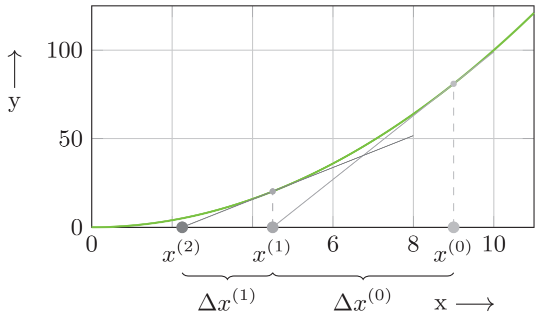


Figure 2.5: Exemplary NR-Method application on $f(x) = x^2$ (based on [34])

In order to use the general NR-Method to solve the power flow equations, the complex nodal voltages of Equation 2.18 must first be divided into their real and imaginary parts using either cartesian or polar coordinates. [34] Correspondingly, the differentiable representation of Equation 2.18 for a n-dimensional vector $\vec{v}$ of unknown voltages is:

$$f\left(\underline{\vec{v}}\right) = \begin{pmatrix} \Re\{\underline{v}_1 \cdot \underline{i}_1^*\} \\ \cdots \\ \Re\{\underline{v}_n \cdot \underline{i}_n^*\} \\ \Im\{\underline{v}_1 \cdot \underline{i}_1^*\} \\ \cdots \\ \Im\{\underline{v}_n \cdot \underline{i}_n^*\} \end{pmatrix} \tag{2.24}$$

Equation 2.21 then becomes

$$f\left(\underline{\vec{v}}\right) \approx f\left(\underline{\vec{v}}^{(k)}\right) + \underbrace{\begin{pmatrix} \left[\frac{\partial p}{\partial e}\right] & \left[\frac{\partial p}{\partial f}\right] \\ \left[\frac{\partial q}{\partial e}\right] & \left[\frac{\partial q}{\partial f}\right] \end{pmatrix}}_{\left[J_f\left(\underline{\vec{v}}^{(k)}\right)\right]} \cdot \begin{pmatrix} \Delta\vec{e} \\ \Delta\vec{f} \end{pmatrix} \overset{!}{=} \vec{0} \tag{2.25}$$

The resulting recursive update rule to solve the power flow equations is given in Equation 2.26.

$$\underbrace{\Delta\underline{\vec{v}}^{(k)}}_{\underline{\vec{v}}^{(k+1)} - \underline{\vec{v}}^{(k)}} = -\left[J_f\left(\underline{\vec{v}}^{(k)}\right)\right]^{-1} \cdot f\left(\underline{\vec{v}}^{(k)}\right) \tag{2.26}$$

$\Delta\vec{e}$ and $\Delta\vec{f}$ describe the real and imaginary parts of the complex nodal voltage vector $\underline{\vec{u}}$. As depicted in Equation 2.26, the update rule in the NR-Method applied for power flow calculations depends on the rebuilding and inversion of the Jacobian matrix in each iteration. While matrix rebuilding is not necessarily computationally challenging, its inversion can become computationally and memory intensive, particularly for large grids. Consequently, the reduction of computational complexity is part of several publications and research activities. Different methods such as the decoupled power flow make use of the mathematically weak coupling of individual power flow equation components, the grid structure or even exploit the sparsity of the Jacobian matrix. [34] For a comprehensive overview, please refer to [42].

Backward-Forward sweep method

While the NR-Method is probably the most popular, general approach to solve power flow equations that can be used for radial and meshed grid structures, depending on the concrete grid structure and the number of nodes, its application may be computationally expensive. For radial grid structures, [44], [45] propose a power flow representation called "DistFlow". Also referring to the equations as "branch flow equations", the authors formulate the power flow problem focusing on the line quantities between the grid nodes. These equations can be solved using a backward-forward sweep approach. [46]

Particularly of relevance to the present work is the general solution algorithm as presented in [46]. Assuming that the source bus that is the slack node is labeled as 0, the voltage set-point at this is v_{set} and that there are no other voltage-controlled buses in the grid, the algorithm proceeds as follows:

1. Initialize all nodal voltage magnitudes at the end buses of the feeder to 1 p.u..
2. Calculate the power flows between all end buses and their next adjacent upstream bus.
3. Perform step 2 until the source bus is reached.
4. Calculate the $v_{\text{mis}} = \left| |v_{\text{set}}|^2 - |v_0|^2 \right|$, where v_0 is the resulting voltage value of the just executed first backward sweep (also referred to as iteration) with counter $k = 0$.
5. Set $|v_0|^2 = v_{\text{set}}^2$. Using the values of the latest calculated power flows from step 2, start from the source bus and calculate the accounting current for the adjacent downstream bus using the corrected nodal voltages of the adjacent upstream bus.
6. For each end bus calculate the power flow missmatch between the end bus and its adjacent upstream bus s_{mis}.
7. Check if convergence is reached: If s_{mis} and v_{mis} are smaller or equal a predefined convergence criterion, convergence is reached and the algorithm terminates. If one or both values are bigger, go back to step 2, increment k by 1 and start another sweep.

2.2.3 Summary

In this section, the fundamentals of electricity grid modeling and power flow calculations are presented. They are essential to determine either a single grid state or time-sequential steady-state electricity grid time series. Following the introduction of the mathematical electricity grid model, different power flow solving methods are presented. Thereby, aspects of two approaches relevant for this thesis, the NR-Method and a backward-forward sweep method, are explained in detail. Due to the inversion of the Jacobian matrix, the “classical” NR-Method has its limitations with regard to scalability. Moreover, its application in the context of an agent-based simulation model is not straightforward due to its centralistic approach. However, combining both approaches result in a viable Multi-Agent System (MAS)-compatible power flow solving algorithm. This is done in Section 4.5 with the presentation of a newly developed Distributed Backward-Forward Sweep Algorithm (DBFS).

2.3 Power System Modeling and Simulation – State of the Art

Time series generation for modern distribution system planning, operation, and analysis requires an electricity grid model, power flow solving algorithms and information about the usage profiles of connected system participants. In addition, co-simulation frameworks can be a valuable complement to investigate sectoral interdependencies resulting from sector-coupling in multi-energy systems. This section presents an excerpt of current models and frameworks for each of these ingredients.

2.3.1 Steady-State Power Flow Execution and Simulation

Power flow simulations models can be distinguished into proprietary or commercial and freely available ones. Well-known proprietary tools for steady-state power flow calculations include INTEGRAL [47], NePlan [48], DIgSILENT PowerFactory [49], PowerWorld [50], PSS®SINCAL [51], Adaptricity.Plan [52] or the envelio Intelligent Grid Platform [53]. Each offers different functions beyond a purely stationary power flow calculation, e.g., the execution of dynamic calculations, short-circuit calculations or decision support. One of their particular advantages is the provision of a graphical, easy-to-use user interface. One of their major drawbacks, in particular from a sector-coupling perspective, is their missing or only minimal availability of application programming interfaces. Furthermore, some tools do not or only partially provide access to the underlying mathematical calculation models. This limits or, to some extent, prevents their integration into a sector-coupled co-simulation or the application of custom scripts.

In addition to proprietary tools, several freely available, partially OS frameworks are suitable to perform quasi-stationary power flow calculations. Two packages that are neither fully proprietary nor entirely usable without proprietary software are the

Matlab-based package MATPOWER [54] and PSAT [55]. While both are in principle OS and freely available, either the commercially available software MATLAB [56] or one of its free derivatives like GNU Octave [57] is required for their execution.

In general, every freely available programming language can be used to implement power flow equation solving algorithms. However, there are already several different implementations in different languages available. For instance, the PYPOWER [58] package, a port of MATPOWER to the Python programming language, provides power flow and optimal power flow capabilities. As a result of the official discontinuation of the development of PYPOWER in 2014, the tools PyPSA [59], pandapower [60] and GridCal [61] were published. While all of them incorporate parts of PYPOWER, each of them took different development directions. [62] Even if not directly related to electric grid power flow calculation, for the sake of completeness pandapipes [63], a piping grid simulation tool, also can be mentioned in this context as it is built on experiences gathered with pandapower. Its main target is the design and operation evaluation of multi-energy grids.

While all tools presented so far are, in general, capable of generating time series by executing them several times, they have not been designed with the primary objective to generate large time series with several thousands of data points, but only for a single or only a few grid operating points. Performance considerations regarding the execution time of power flow calculations are only of minor interest in any of these tools, if at all. In contrast, a few freely available, OS tools have specifically been developed and designed for time series generation purposes.

For example, the freely available Open Distribution System Simulator (OpenDSS) of the Electric Power Research Institute (EPRI) has been designed and developed specifically with time series generation in mind. [64], [65] There is also a high-level Python interface, PyDSS [66], available. A second example is the power distribution system simulation GridLAB-D™ [67], [68]. Being close to the model presented in this thesis, GridLAB-D™also uses an agent- and information-based model approach that allows for the decomposition of the overall grid power flow problem into subproblems. System participants are modeled using dynamic differential equations and it provides control and market mechanisms out of the box. Both tools, OpenDSS and GridLAB-D™, provide models which allow for an endogenous simulation of system participants and their grid usage. Additionally, both of them are designed and implemented using the concept of actors, which allows their application even on large grids. Similarities to GridLAB-D™and OpenDSS are also apparent in the model presented in [9], [10], which partly provides the theoretical basis for the present work. The respective system participant models are provided endogenously in both of them. Accordingly, these models are used to determine system participant grid usage and subsequently the quasi-stationary grid operating point to generate several different time series. However, in contrast to GridLAB-D™and OpenDSS, the approach of [9] and [10] does not use an actor- or scalable concurrency-based implementation, which prevents its application on large grid. It even uses a discrete-time advancement mechanism instead of a discrete-event one, resulting in performance bottlenecks already on small grids.

2.3.2 System Participant Grid Usage Profiles

If system participant feed-in or consumption patterns are not endogenously provided in a simulation model, they must be provided exogenously. There is a variety of different data sources and models for almost every system participant asset available. A classic source for household load profiles is the consideration of standard load profiles. [69] However, these profiles are only valid for a large number of participants and therefore may not be used for single households. As already mentioned, it is also possible to use field measurement data, if available. Such kind of data, however, may pose additional challenges due to data errors or missing values.

The Renewables.Ninja (RN) platform provides another source of exogenous feed-in data for renewable assets. [70], [71] Besides the provision of pre-calculated WEC and PV time series, the platform also provides the underlying models used to generate the time series. These models can be seen in conjunction with other publications like [72]–[74] which all provide different approaches to generate synthetic system participant usage patterns. If benchmark grids instead of real-world grids are used for analysis purposes, like the SimBench data set [12], they sometimes also include operating point information or system participant usage pattern. [75]

2.3.3 Co-Simulation Frameworks

Co-Simulation frameworks allow for a coupling, synchronization and data exchange of multiple independent simulation models. They can be used for several application cases, including the execution of sector-coupled simulations, coupling of system participant or power flow simulations or considering communication infrastructure as part of distribution system simulations. Examples for such frameworks are the *Framework for Power System and Communication Networks Co-Simulation* (FNCS) [76], *mosaik* [77], [78] and the *Hierarchical Engine for Large-scale Infrastructure Co-Simulation* (HELICS) [79]. The main target of FNCS is a power grid and communication network simulator. It provides the ability to co-simulate transmission and distribution power grid simulators with a communication network simulator. With similar capabilities but the greater target to simulate full smart grid infrastructures, mosaik may serve as a testbed for cyber-physical systems strategies. In contrast to FNCS, mosaik specifically focuses on the distribution grid level and provides several adapters to different frameworks. [80] HELICS, a co-simulation engine developed within the Grid Modernization Laboratory Consortium, also targets the field of cyber-physical-energy co-simulation. It provides interfaces for many different grid simulators like MATPOWER, GridLAB-D™, OpenDSS and others. A recent application case of HELICS is its utilization in the Grid-Enhanced, Mobility Integrated Network Infrastructures for Extreme Fast Charging (GEMINI-XFC) project. In this project, a large scale agent-based EV simulation is coupled with a grid simulation in order to develop control algorithms to minimize the impacts of EV charging and support EV grid integration at scale. [81] All presented coupling frameworks are freely available under different OS licenses.

2.3.4 Summary

This section presents the state of the art of different approaches and systems for power system modeling and simulation for time series generation. When considering time series for grid planning, operation and analysis purposes, the most crucial part is suitable tools. Therefore the section broadly outlines different proprietary and freely available steady-state power flow solving tools. While built with the same aim, almost all of them differ in terms of their theoretical approaches.

Furthermore, different tools and methodologies to generate system participant usage profiles are presented. They are the second crucial ingredient required by all power flow solving tools that do not have endogenous system participant models but rely on externally provided data.

The third part that is outlined and plays a crucial role for multi-energy systems at the distribution grid level is co-simulation frameworks. Within the section, three state-of-the-art co-simulation frameworks for multi-energy system simulation are shortly introduced. However, even if there is already a large variety of different approaches and tools available, none of them is currently able to address all requirements of modern active distribution system simulation for planning, operation and analysis purposes. This situation is discussed in detail in chapter Section 4.1 and as a consequence, a newly developed simulation model, SIMONA, will be presented.

3 Modeling and Simulation Fundamentals

Different methods are available to study the impact of changing conditions on the electric power system. Their applicability is usually linked to several different influencing factors, such as the specific scope of the study, the maturity of new technology, or the economic risk in case of failure. Thus, investigating the impact of a new control algorithm in the context of a small-scale field test may make sense, as the financial risk in case of an unpredictable event is limited. However, such a field test may not allow conclusions to be drawn about possible interdependencies or novel effects in a large-scale application of the control algorithm. Moreover, it is often impractical or even impossible to evaluate all potentially possible cases in the context of a physically conducted field test and evaluate the corresponding effects. [82] Particularly in the area of critical system infrastructure, whose failure can have significant consequences for society, conducting experiments on the running system is only possible to a minimal extent, if at all.

Consequently, not only in the context of energy system research but also in a large number of other research areas, an increasing trend towards a simulation-driven investigation of increasingly complex systems can be identified. [83]

In the following chapter, the basics of modeling and simulation required to support the understanding of this thesis are briefly introduced. After differentiating between the terms modeling and simulation, this chapter continues with an overview of different model and simulation types. Furthermore, a classification of simulations based on different time concepts is given. Thereupon, the theory of Discrete-Event Modeling and Simulation (DES) is introduced, representing one available approach to represent time within a simulation model. This introduction includes additional remarks on different execution mechanisms of DES. The last part of the chapter introduces the concepts of Multi-Agent System (MAS) and Agent-based Simulation (ABS). Within this section, the theoretical foundations are given, and specific implementation approaches are discussed.

3.1 Basics of Modeling and Simulation

While the terms *modeling* and *simulation* are often used synonymously, they describe two different, most of the time subsequent process steps in the context of system research and analysis. Starting with the model development followed by its application within a simulation, this process allows the investigation of the behavior of complex systems in different scenarios as well as to perform what-if-analyses. The fact that experiments can be carried out repeatedly with changed input parameters and the behavior of a system over a long time horizon can be simulated faster than time passes by in reality makes modeling and simulation a widely used and accepted methodology in science and practice. [84] It is crucial to carefully design models used within a simulation to ensure that the simulation outcome provides deeper insights and understandings of real-world challenges. While it is neither possible nor economically efficient to have a model representing all facets of the real world, the applied model must capture at least all required features of the real world to be *valid*. [85] The following chapter gives a brief overview of the most important concepts and definitions of modeling and simulation.

3.1.1 Modeling and Simulation Concepts

In order to investigate a complex, real system, first of all, it must be clarified what the term *system* means. While [85]–[89] provide several slightly varying definitions of a system, they all have in common, that a *system* can be described as a *structured* collection of *entities*. The structure is defined and formed by interactions or interdependencies between these entities, e.g., entities might interact with each other to fulfill some logical goal. There are different ways available to study such a system (Figure 3.1). The particular way to choose depends on the existing system of interest and the availability of its manipulation parameters, e.g., the possibility to experiment with the existing system and the reproducibility of experiments. In particular, for critical systems or when performing experiments for multiple scenarios is required, using the existing system may not be suitable. In that case, using a model, may it be physical or mathematical, is the preferred way to go. Although these models are usually only a simplification of the existing system, they support flexible system investigation and, if correctly modeled, their results are valid for the existing system. Most of the time, physical models are more expensive to build than mathematical models while at the same time may be more limited in their use.

Whenever *simulation* is chosen as a suitable approach to study a system, as depicted in Figure 3.2, the process of such a study is a tightly coupled and iterative process. Depending on the granularity of description, this process consists of multiple repeated steps. The first step is always the analysis of the system to be investigated, followed by the *modeling* or *model design* based on the insights gained from this analysis. Subsequently, after its *implementation* the model is executed within a *simulation* using a *simulator*. The generated *results* are then analyzed again within

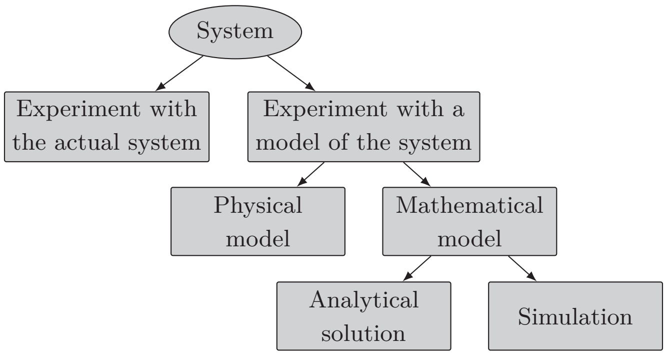

System
Experiment with the actual system
Experiment with a model of the system
Physical model
Mathematical model
Analytical solution
Simulation

Figure 3.1: Ways to study a system (based on [86])

an *execution analysis* which results in *insights* of the simulated system. These *insights* can then be used again to verify and validate the model as well as to refine it, resulting in an improved *model* which can be executed again. Sometimes, depending on the insights gained from the result analysis, it also may turn out that the model is valid, but mistakes in the actual implementation have been made. In that case, only the implementation is adapted and the model is re-executed. [88], [90] It is important to mention that the term *simulator* does not allow for any conclusion on its specific properties. Consequently, a *simulator* can represent both a physical as well as a digital component for *model execution* in the context of a simulation. [85] Within the scope of the present work, all developed models are executed using a digital, computer-aided simulator. Therefore, for all subsequent descriptions, it is assumed that a computer as a simulator is used.

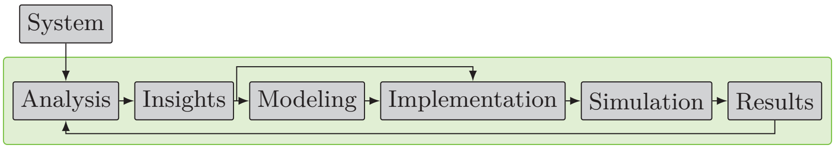


Figure 3.2: Modeling and simulation cycle (based on [90])

Before the actual modeling can take place, the first two steps when building a simulation model are *analysis* of the existing system followed by a derivation of the required *insights* to build a suitable model. The subsequent *modeling* is the creation of a physical or digital abstract representation of the actual system. While a model tries to capture the whole system it represents, it is usually more straightforward than the existing system under investigation. The amount of simplifications made depends on the specific interest of the analyst using the model. On the one hand, the model should incorporate all essential features to allow for investigations of interest, while on the other hand, it should not be too complex to prevent the interpretation of its outcome. Whenever a digital model is used, it can be described either mathematically or in any other form of specification suitable for the specific application. [82] Depending on the complexity of the model, it may consist of or contain several *entities* representing components, objects, or specific submodels in the system that require explicit representation. [84] In order to ensure the suitability of a model for the concrete investigation case, the *verification* and *validation* of a developed model are of particular importance in the modeling and implementation steps. Verification can be done for causal models, e.g., by varying the input data and observing the output data. The model's functionality is verifiable by comparing the model's behavior with its specifications. After its verification, a model can also be validated. A validation is done by using input/output data of either a real world system or a similar, already validated, model. The existing input/output data is than compared with the input/output data of the newly developed model that should be validated. This process makes sure the developed model's behavior and characteristics correspond to the modeled system's behavior and characteristics to fulfill the intended model objectives. [82], [90]

When a model of a system is available, the *simulation* is the actual operation of the model - it is the process of using a model to study the behavior of the system of interest. [82] Normally, this simulation process is carried out under well-defined test conditions with changing parameters to investigate and learn from the different behaviors and outcomes of a model under different conditions. More precisely, the execution and investigation of models within a simulation follow a *learning-by-doing principle*. [88] In order to execute a simulation, be it a physical or a digital one, a *simulator* is required. This simulator is the concrete representation of the abstract model. Depending on the application case, the simulator can either be a physical, a digital or a hybrid concrete instance of the model. The transformation process, which turns the abstract model into a concrete simulator, is called *implementation*. Depending on the model's abstract definition, the specific implementation may differ in its design but fulfills all required attributes specified by the model.

3.1.2 Types of Models

The need for models to study systems to understand and improve them has led to many different modeling approaches. They are roughly divisible into the two categories of bottom-up and top-down models (Figure 3.3). This categorization is valid in general but also specifically for energy and power system models. [91], [92]

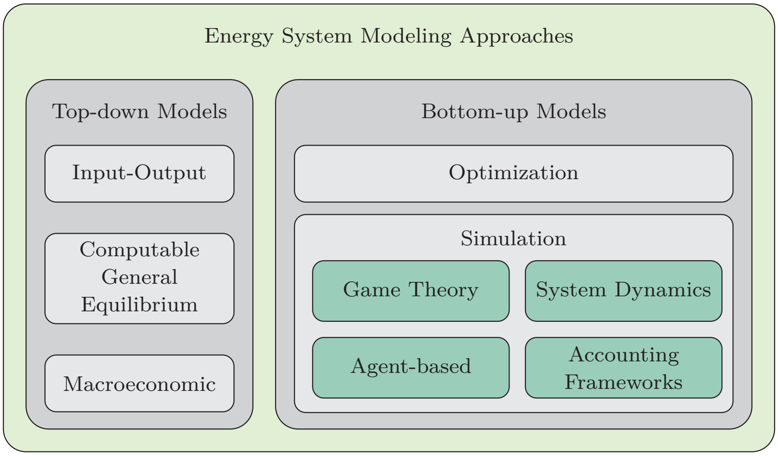


Figure 3.3: Power system modeling approaches overview (based on [10], [92])

Top-down approaches describe a system from a macro or global perspective. Here, single entities, their individual properties, and local phenomena of the system are aggregated to one or a small set of variables that describe the overall system. Examples of top-down models are Input-Output (I/O) models, Computable General Equilibrium (CGE) models and general macroeconomic models. [92] A more illustrative example in the field of power system models is the investigation of the transmission grid. A way to simplify its calculation is to aggregate several distribution grids at one single coupling node on the transmission grid level. While this approach is feasible to investigate different aspects of the transmission grid level, it might neglect potential interdependencies arising from the interaction between the transmission and the distribution grid layer, as the distribution grid is only modeled in an aggregated instead of an explicit fashion. However, even if the aggregation neglects certain interdependencies that might occur between the different grid levels, the actual simulation results of a top-down model can be disaggregated again.

The inverse of the previously applied aggregation function has to be applied to the results to achieve this. Depending on the specific use case, a disaggregation might be sufficient enough to provide the required system insights.

Bottom-up approaches describe a system from the micro or local perspective. They comprise detailed descriptions of single entities and their individual technical or economic attributes. Examples of bottom-up models are optimization models or simulation models. Optimization models try to find a solution for a given problem by minimizing or maximizing an objective function. The assumption inherent in such models is that the overall system tends to end up in a perfect or at least partial sufficient equilibrium state. [92] Furthermore, in many models, perfect knowledge about the future or at least the whole simulation horizon is assumed. [13] may serve as an illustrative example for the application of optimization models for power system research. Another class of bottom-up models are simulation models, not to be confused with the already defined term *simulation* (see Subsection 3.1.1). While a *simulation* is defined as the actual execution of a model in general, a *simulation model* also represents a specific model class. However, the naming of this model class is closely connected to its execution process. Simulation models are explicitly designed to study the behavior of the system of interest without relying on a closed equilibrium framework and perfect anticipation of the future. [93] They are rather constructed by a set of rules or technical descriptions that define the interdependencies and single processes within the model. [94] This approach allows for a step-by-step computational model development process which makes it applicable to a broad range of systems and models - from elementary system representations up to very complex ones. [92] Furthermore, they can be applied to determine solutions for realistic scenarios where only limited information is available. [10] The broad range of simulation models allows for further differentiation between them. According to [92], Accounting Frameworks (AF), Game Theory (GT) models, System Dynamics (SD) and Agent-based Models (ABMs) belong to this category.

3.1.3 Types of Simulations

The versatile applicability of computer simulations for the investigation of complex systems has led to the emergence of many different simulation paradigms, formalisms for simulation descriptions, and a multitude of different implementation options. Depending on the study's objectives and the characteristics of the system of interest, a suitable simulation can be chosen. Accordingly, there are different taxonomies and categories to classify simulations (for a broad overview, see [90], [95], [96]). One way to categorize them is their internal representation of time. Additionally, as the representation of time also plays a significant role within the discrete-event simulation environment developed in this thesis, different types of simulations are described in the following based on their time representation. In order to enable a coherent understanding of the concept of time, first of all, the different ways to represent temporal dependencies in a simulation model will be discussed. Subsequently, the actual classification and description of different simulation types will be provided.

[85] distinguishes between two different notions of time within a simulation, *physical time* and *simulation or logical time*, both of them being either locally or globally available. *Physical time* represents the commonly known metric time or wallclock time and is measured by physical clock ticks. *Logical time* in contrast is measured by ticks of a clock embedded in a model. Global time means that all model entities operate on the same time base. Locally means an entity has its internal time base. [85] In addition to these two notions of time, a third notion of time, the actual *real world wallclock time*, can be defined. This time refers to the actual simulation's processing time, e.g., how long its execution takes in real-world physical time.

Based on the preceding explanations, the most common simulation types differentiated by their notion of time are listed below.

Steady-state or static simulation No physical nor logical time is known in a steady-state simulation. Within such a simulation, a model could represent a system where a time-independent equilibrium state needs to be found. An example is the simulation of one specific grid usage case, e.g., heavy load case or light load case.

Dynamic Simulation A dynamic simulation has either a physical or logical time representation that evolves over time. A distinction can be made between continuous and discrete dynamic simulation.[3]

Continuous Simulation Within this simulation, time may be represented physically or logically on a global or local base. The update interval between two different time steps is regularly, may be arbitrarily small (represented as a real number), and the model's state changes continuously. A set of differential equations normally represents this kind of simulation. [8]

[3]The term *dynamic* concerning the representation of time in simulation models may not be confused with the commonly used but unprecise term *dynamic power system simulation* which refers to the simulation of transient processes in the electricity grid. This kind of simulation is most of the time performed using models of differential equations. Therefore, *dynamic power system simulation* is equal to *continuous simulation* in terms of modeling and simulation theory.

Discrete Simulation Time advancement is assumed to change only at a discrete set of points in time. It can be further distinguished between a fixed time increment or an event-driven time advancement mechanism.

Fixed Time Increment Driven or Discrete Time Time may be represented either physically or logically. The update interval between two subsequent points in time is regularly and with a fixed increment of time. The simulator repeatedly executes the system model, with each repetition advancing the simulated time in a fixed defined time interval.

Event Driven or Discrete Event Time may be represented either physically or logically. The update interval between two subsequent points in time may be irregular or regular and an update is assumed to happen only at discrete points in time. The simulator maintains a centralized or decentralized event schedule and time elapses from one event to another while each event corresponds to a local or global model state change.

Hybrid Simulation A combination of discrete and continuous time variables within the same simulation.

When implementing a model in a simulator, not every model is suitable for each of the mentioned simulation types. Accordingly, selecting the appropriate simulation type must be adapted to the respective model and its concrete design.

3.1.4 Summary

This section briefly outlines modeling and simulation fundamentals to introduce the reader to the basics required to understand the following chapters. Furthermore, the different notions of time within a simulation are provided, and several simulation types are introduced based on a classification of their time representation. For the remainder of this thesis, the following terms will be used accordingly:

System A system is a real existing or imagined construct, which is to be represented in a modeling context.

Model A model is an abstract specification or representation of a unique system of interest. It may consist of simplifications of the real system but covers all required properties of interest. Based on a verification and validation process, the model can be used within a simulator to reproduce the relationship between input and output data of the system. It can be distinguished between global perspective top-down models and local perspective bottom-up models.

Entity An entity is any component or object within the modeled system that requires an explicit representation. It can also be considered to be a "model inside a model" or a submodel.

Implementation The process of transferring the abstract model into a concrete simulator is called implementation. Within this thesis, implementation always refers to transforming the abstract model description into executable software code that can be compiled and executed as a computer-based simulation.

Simulator A simulator is the concrete representation of one or multiple models, and it represents the execution environment to execute the models of the simulated system.

Simulation A simulation is the process of actual operation or execution of the model. Within this thesis, the term simulation always refers to a computer-aided or computer-based model execution.

Physical Time Represents the metric or wallclock time, measured in physical clock ticks.

Simulation or Logical Time Represents time by ticks of a clock that is embedded in a model, while the model ticks may differ from the physical clock ticks.

Wallclock or Execution Time The actual physical time of the model execution. That is, the time the simulator is executing the model represented in physical time.

3.2 Discrete-Event Modeling and Simulation

Within a simulation, time can be represented in different ways. In particular, for a dynamic simulation, which is a simulation where time is not static but advances, different approaches are available (see Section 3.1.3). As a reminder, it has been shown that time in terms of a dynamic simulation model can be either continuous[4], fixed time increment driven[5] or event-driven[6].

[4]Continuous means regular update interval between two time steps. The update interval may be arbitrarily small. This simulation type is frequently modeled as differential equations.

[5]Fixed time increment driven means discrete time steps with regular update interval between two points in time.

[6]Event-driven means discrete time steps with regular or irregular whenever an event happens.

The event-based approach assumes that the represented overall system only changes when a concrete activity, an *event*, takes place. If there is no event, the system is in equilibrium and can be described unambiguously. When an event occurs, the system changes and is in a different state after the event processing. This representation of time within a simulation model has some significant advantages, which is why it is also used within the simulation model developed in this thesis. First, an event-driven simulation tends to be more efficient than fixed time increment-driven approaches as points in time with no activity are neglected. [85] Secondly, as [90] points out, it allows for comparably simple integration of uncertainty and stochastic behavior of entities within the simulation model.

This chapter will provide a brief overview of the required fundamentals of DES in order to support the understanding of the developed simulation model in this thesis. For a comprehensive theoretical overview of different DESs approaches, please refer to [85]. [84], [97] provide detailed introductions from a practical perspective.

3.2.1 Classic Discrete-Event System Specification

A Discrete-Event System Specification (DEVS) is a way to describe a system in terms of system theory. In system theory, the concept of *composition* and *decomposition* describes how a system may be coupled together or broken down into components. Each of these *components* may have a *behavior*, which is the outer manifestation of the component, and a *structure*, which is the inner constitution of the system. With these concepts, system theory is assumed to describe a system closed under composition, which means that the outer manifestation and the system's structure are again describable using the original system theory terms. This ability to compose larger and larger systems from previously constructed components simultaneously results in a hierarchical construction of the resulting system itself. This idea of component composition leads to an overall system with a well-defined structure and behavior. Moreover, it also makes it highly modular because each component may interact with another component using input and output ports that describe their behavior. [85] This understanding of systems is of particular importance when a coupled system of several DES components is modeled.

A second important concept within DES is the differentiation between external and internal events. Internal events are always pre-scheduled based on the time the model intends to stay in its current state if not interrupted by any other event. This time may be 0 (transitory state), any real number or ∞ (passive state). Additionally to these internal events, a DES may be able to receive and process external events.

External events are never pre-scheduled and may be sent to the DES from another DES within a coupled DES or any other entity that can be imagined.

With these two concepts at hand, the general formal description of a single or *atomic* DES can be introduced. The following formal description is adopted from the classic DES specification of [85]. The term *classic* in this sense is used in [85] to distinguish different standard DES approaches presented. However, as this thesis only uses one approach for single DES the *classic* term will be neglected from now on.

According to [85], a DES model M can be described formally as a structure

$$M = \langle \mathbf{X}, \mathbf{Y}, \mathbf{S}, \delta_{\text{ext}}, \delta_{\text{int}}, \lambda, ta \rangle \tag{3.1}$$

where

$\mathbf{X}$	is a set of input values,
$\mathbf{Y}$	is a set of output values,
$\mathbf{S}$	is a set of states,
$\delta_{\text{int}} : \mathbf{S} \rightarrow \mathbf{S}$	is the internal transition function,
$\delta_{\text{ext}} : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbf{S}$	is the external transition function,
$\lambda : \mathbf{S} \rightarrow \mathbf{Y}$	is the output function,
$ta : \mathbf{S} \rightarrow \mathbb{R}^{+}_{0,\infty}$	is the time advance function.

The total state set $\mathbf{Q}$, required to create a function of input values $\mathbf{X}$ to model states $\mathbf{S}$ using the time elapsed e since the last state transition is defined as

$$\mathbf{Q} = \{(s, e) | s \in \mathbf{S}, 0 \leq e \leq ta(s)\} \tag{3.2}$$

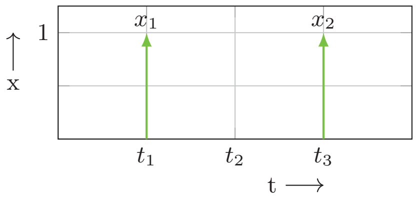
With the formal description, the actual model execution using an execution mechanism (see Section 3.2.3) can be described generically. The modeled system is always in a state s. It remains in this state for time $ta(s)$ as long as no external event occurs. When the resting time expires, that is if $e = ta(s)$, the corresponding output value $\lambda(s)$ is issued *before* the system changes to state $\delta_{\text{int}}(s)$. When the system is in a state s for an elapsed time e, such that (s, e), and an external event $x \in \mathbf{X}$ occurs at a time where $e \leq ta(s)$, the state transition is triggered and the system changes to the defined state $s' = \delta_{\text{ext}}(s, e, x)$. Thus, the internal transition function $\delta_{\text{int}}(s)$ defines the system's next state s' if no external events occur since the last transition and the external transition function $\delta_{\text{ext}}(s, e, x)$ defines the system's next state s' if an external event occurs. The latter is determined using the received input event x, the current state s and the time the system has been in this state e when the external events is received. In every case, the result is a new system state s' with a new state time $ta(s')$. This process is repeated until a termination condition, e.g., reaching the last simulation time point, is met.

Figure 3.4 illustrates this process with a short example. After its initialization, the model is in state s_0 with the corresponding time advance function $ta(s_0)$. When the input event x_1 is received, it triggers a state transition from s_0 to s_1, the elapsed state time is reset to zero and $ta(s_1)$ becomes the new time advance function. It is important to mention here, that no output event y is emitted because when x_1 is received, $e \leq ta(s_0)$ holds true. Being in s_1, the expiration of the resting time e, that is when $ta(s_1) = e$, the output function $\lambda(s_1)$ is executed resulting in the emission of the output event y_1 subsequently followed by a state transition from s_1 to s_0 with the time advance function $ta(s_0)$ becoming active again. Figure 3.5 illustrates the transition process inside the model between t_1 and t_2. Back in s_0 again, an input event x_2 is received at t_3, again $e \leq ta(s_0)$ holds true, resulting in another state transition from s_0 to s_2.

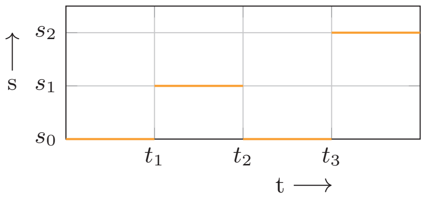
With Equation 3.1, the model of a DES is fully described. However, it does say nothing about its actual execution and may raise questions about how events are stored and organized or on their termination conditions. In order to execute a DES, different mechanisms are available, which are outlined in Section 3.2.3. Before the model execution mechanisms are introduced, in the next part, a coupled DES network is introduced, which allows for the composition of several DES for more complex application cases.

3.2.2 Discrete-Event Specified Network

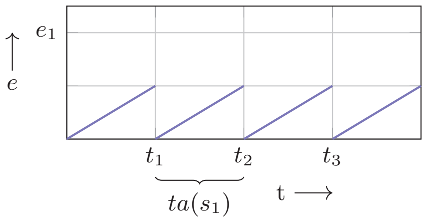
With the atomic DES formalism at hand, the corresponding *coupled* formalism can be introduced. Coupled in this context means that several atomic DES components are coupled together in order to form a new, more complex system. One way to describe such a coupled system is the Discrete-Event Specified Network (DEVN) formalism, also called Modular Coupled Discrete Event System. In a DEVN, components can neither access other components state or internal variables nor influence other components directly. Instead, any interaction with other components has to be done by exchanging messages. Furthermore, the actual interaction between components is clearly defined along the coupling points. As a result, output events generated by one component are transmitted along the couplings to the input interface of another component, where they cause external events and may trigger state transitions at the receiving component. [85]



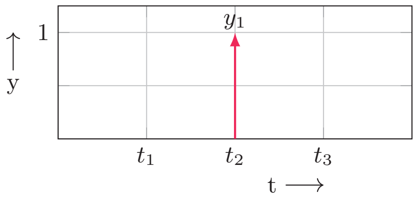
(a) Trajectories of input events $x \in \mathbf{X}$



(b) Trajectories of states $s \in \mathbf{S}$



(c) Time advance function $ta(s)$ Trajectories



(d) Trajectories of output events $y \in \mathbf{Y}$

Figure 3.4: Exemplary DES trajectories during model execution (based on [85])

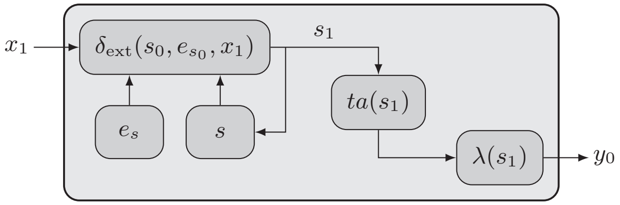


Figure 3.5: DES model state transition between t_1 and t_2 in Figure 3.4

A coupled DEVN N is a structure which, according to [85], can be formally described as

$$N = \langle \mathbf{X}, \mathbf{Y}, \mathbf{D}, \{M_d\}, \mathbf{I}_d, Z_{i,d}, Select \rangle. \tag{3.3}$$

$\mathbf{X}$ is a set of input events, $\mathbf{Y}$ is a set of output events, $\mathbf{D}$ is a set of component references and $Select : 2^D \rightarrow \mathbf{D}$ with $Select(\mathbf{E}) \in \mathbf{E}$ is a tie-breaking function that is used in case of equal next event times occuring in more than one component. $\mathbf{E}$ in this case is the set of all possible events.

For each $d \in \mathbf{D}$, M_d is a classic DES model according to Equation 3.1. The components' influence on each other, this means, being able to trigger an action of another component, is defined by the influencer set $\mathbf{I}_d$ defined $\forall\, d \in \mathbf{D} \cup \{\mathbf{N}\}$ as

$$\mathbf{I}_d \subseteq \mathbf{D} \cup \{N\}, d \notin \mathbf{I}_d. \tag{3.4}$$

Additionally, each component $d \in \mathbf{D}$ incorporates a function $Z_{i,d}$ that maps the events of each influencing component on its interfaces. In other words, $Z_{i,d}$ is required to translate an input of an influencer i to the output of d and vice versa. $Z_{i,d}$ is defined as

$$\begin{aligned} &Z_{i,d} : \mathbf{X} \rightarrow \mathbf{X}_d, \text{ if } i = N. \\ &Z_{i,d} : \mathbf{Y}_i \rightarrow \mathbf{Y}, \text{ if } \ d = N. \\ &Z_{i,d} : \mathbf{Y}_i \rightarrow \mathbf{X}_d, \text{ if } d \neq N \text{and } i \neq N. \end{aligned} \tag{3.5}$$

A coupled modular model can be fully described with Equation 3.3–3.5. However, again this description says nothing about the actual execution of a DEVN. Different ways that enable DES model execution will be presented in the following section.

3.2.3 Discrete-Event Execution Mechanisms

In order to execute a DES, different execution mechanisms are available. As mentioned earlier, DES are closed under coupling, and the same execution mechanisms can therefore be used for both formalisms presented above. In general, the execution mechanism, also called DES implementation or DES simulator, is an algorithm that defines how events are scheduled and processed. Remember that a DES is a simulation model in which time only changes when an internal or external event occurs. This fact leads to the conclusion that an execution algorithm must ensure the correct processing of events. Furthermore, most of the time, the actual implementation of an execution mechanism includes one or more simulation executive or control entities that maintain a list of events and may have a partial or complete overview about all currently available events that may change the system state. [85]

In principle, an individual execution mechanism can be developed for a particular DES. However, four common approaches already exist, which may be used directly or serve as a blueprint for an individual execution mechanism. In the following, each of them is briefly described. For a detailed description and mechanism comparison, please refer to [98] and [99], respectively.

Three-Phase Method In the three-phase method, firstly presented in [100], events are classified into b-events and c-events. B-events, from *bound* or *booked*, are events that are scheduled to occur at a pre-defined point in time. C-events, from *conditional*, are events or state changes that depend on the actual condition of a model. Furthermore, the scheduling process is divided into three phases, an a-, b- and c-phase. The execution always starts in the a-phase, where the time of the next event is determined based on the smallest time of occurrence of all b-events and the simulation time is advanced to this time. Afterward, all b-events eligible for this point in time are executed in the b-phase. In the c-phase, all conditional c-events are executed, if any. Afterward, the execution process starts again in the a-phase or terminates of the pre-determined end time is reached. [97]

Event Scheduling In general, this approach is close to the three-phase method. The key difference is that there is no c-phase nor are there c-events. The c-phase is replaced by an “event routine” which specifies all actions that follow an event or state change. This approach has the advantage that there is no need to actively look for new c-events every time, making it computationally efficient. However, it comes with the drawback that all possible consequences of an event need to be considered during implementation, making the implementation process quite challenging. [97]

Activity Scanning In contrast to event scheduling, where only the a- and b-phase of the three-phase method exists, this approach only consists of a single c-phase and all events are treated as c-events. Within the execution process, all available events are scanned again and again to determine which events can be executed at the current time. If no events can are executable, the simulation time moves on until the next smallest time of all events is reached. The fact that a continuous scanning of all events is required in this approach makes it computationally challenging. In particular, it is not suitable for models with a large number of entities and events. [97]

Process-based In the process-based approach, the main building blocks are not events but processes. A process is assumed to be a sequence of activities an entity must execute. Whenever a delay in a process is identified, may it be conditional or unconditional, the execution mechanism must ensure that entities can continue their process. This can be done by either activating the progression at the end of an unconditional delay or removing a conditional delay if all required conditions have been met. [97] Using a process-based approach introduces additional requirements regarding the implementation, which imply the possibility to suspend and delay process execution, as well as their continuation at a later point in time. [101] As pointed out in [102], these requirements impose additional challenges when implementing this scheduling approach using generic programming languages.

3.2.4 Summary

This section gives a short introduction to the representation of time within a simulation model using discrete events. Starting with the formal description of a single DES, the formal description of a coupled DES network, specifically a DEVN, is introduced subsequently. In addition, different available model execution mechanisms are presented and briefly described. In the following section, MAS and ABS will be introduced. Having the provided fundamentals at hand, it will reveal that DES and ABS have some conceptual ideas in common, which makes DES a viable approach representing time within an ABS.

3.3 Multi-Agent Systems and Agent-Based Simulation

The theory of agents and subsequently the development of MASs and ABSs has arisen from the field of distributed artificial intelligence. Originating from formal logic and computability theory, the application field of artificial intelligence included particular problems associated with high computational complexity. [103] As a consequence of this complexity, a research area emerged that deals with parallel artificial intelligence and distributed problem-solving. Distributed problem-solving in this context means the decomposition of a complex problem into its individual parts. With appropriate coordination of the resulting parts, the problem can be solved in a correspondingly decentralized and parallel manner. [104] Since a variety of real-world technical systems are composed of either several subsystems or functions, the methodology of solving problems in a distributed fashion has gained popularity in many different research fields. However, the very flexible and wide range of applications of this type of distributed problem solving has led to different scientific discussions since the 1980s, but up until now, no uniformly accepted definition of the individual components of the system exists. [105]

In the following, a short overview about the basics of MASs and its main ingredients, *agents*, the *environment* and their *interactions* is given. Afterwards, the differences between MAS and ABS are outlined. The section ends with a brief outline on the different options available to implement an ABS.

3.3.1 Multi-Agent-Systems

Due to their flexible applicability and the increasing need to solve complex problems in a parallel and decentralized way, MAS enjoy great popularity in different scientific and practical disciplines. [106] Despite or because of this popularity, the concrete understanding of which properties define an agent and how an MAS is composed varies. Until today, there is no universal agreement on the precise definition of an agent, MAS or ABS, whereby available literature controversially discusses, in particular, the term *agent*. [106]

However, there is a broader common understanding about the general components of a MAS. In its basic form, each MAS consists of at least two *agents*, an *environment* in which the agents are located in and which they can perceive and interact with, and the possibility to *interact* with and *perceive* other agents (Figure 3.6).

In the following, all three components of an MAS will be briefly explained. A detailed presentation of the extensive subtleties and details of MAS is deliberately

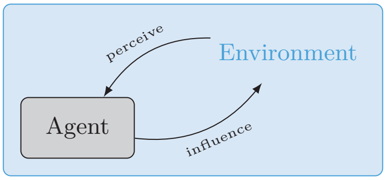


Figure 3.6: General MAS structure from a single agent's perspective

omitted. For more detailed insights into the theory of MAS, please refer to [105], [107], [108]. Furthermore, an up-to-date overview about the history of MAS and the application of agent systems is provided in [109].

Agent

The central and most important ingredient of a MAS is the entity that constitutes it: the *agent*. As the name already indicates, a MAS consists of at least two, but usually significantly more, agent entities. Although no universal definition of an agent exists and it seems to be unlikely that it ever will for a variety of reasons, one of the most popular and also most general definitions is presented in [105]:

"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives."

While this definition has gained some popularity in the MAS community, it still does not take into account the specific requirements of different disciplines which apply MAS techniques. Furthermore, it raises questions about the exact meaning of terms like *computer system*, which can be hardware or software, or *autonomous* which is not clearly defined in [105].

A more recent attempt to define agents is undertaken in [106]. Based on existing MAS used in different disciplines, four different definitions for agents and MAS in general are presented. The classification of agents is based on their capabilities in terms of *individuality*, *behavior*, *interactions* and *adaptability*. *Behavior* at this point may not be confused with behavior from a DES perspective. Even though both terms describe the externally visible manifestation of a component in DES terms or an agent in MAS terms, the behavior of an agent may be significantly more complex than the one of a DES component. The respective definitions are based on complexity classes starting from very simple to very complex agents. [108], [110]–[113] provide a broad picture of the discussions around the specific definition of an agent as well as other definitions and reasonings about agent attributes.

Reducing both the general definition from [105] and the more recent definitional approaches from [106] to the essentials, it can be stated that an agent is either hardware, a software or a combined hardware-software entity, which has sensors and actuators that can perceive and influence its surrounding environment. Furthermore, even if their concrete implementation may vary, an agent's essential properties can be summed up according to [8] and [114] to the following expressions:

Situatedness and Locality The agent is situated in an environment that can either be wholly or partly perceived, and the agent can further interact with it. Locality means that the agent might only perceive and interact with something close to it. Close in this context means that either the geographical distance between two agents or an agent and parts of the environment play a role or that the agent can only perceive and interact with specific other agents but not with all of them.

Autonomy The agent can determine its actions to fulfill its goals by itself without being influenced from the outside. The only influence that is possible is what it perceives from the environment. However, the actual goals on which an action is chosen based on the sensed environment cannot be altered.

Pro-Activity and Reactivity An agent might react to trigger from the environment and take action on its own if fulfilling its goals requires so. Furthermore, if any impulses are coming from the environment or another agent to a particular agent, it may adapt its behavior and react flexibly to handle the new situation.

Rationality An agent always tries to reach its objectives, which means it normally would choose an action that maximizes the probability of reaching its target.

Sociality Any agent is able and interested in the communication with other agents within the MAS.

The multitude of different understandings of agents is both an advantage and a disadvantage at the same time. Its advantage is clearly that agent theory can be applied to a wide variety of problems. However, a decisive disadvantage is that it must be made very clear which form of agents is involved when presenting agent-based approaches. Frequently, controversial discussions arise as to whether these are still agents or merely decentralized software programs.

Environment

In general, an environment in MAS terms is everything a single agent interacts with. From an agent's perspective, the environment influences and sends stimuli and can, in return, be influenced by the single agent entity. The stimuli that a single agent perceives might come from other agents, which are part of the environment from the single agent's perspective or any other passive entity or component. In return, the particular agent's impact on the environment might influence another agent, a group of other agents or any passive entity the agent can reach with is actuators.

In addition to this relationship between a single agent and the environment, the Foundation for Intelligent Physical Agents (FIPA) defines general MAS specifications which mention different entities that may be part of a MAS of software agents. [115] Even though they have not been updated for several years now, they are of particular importance until today. According to the FIPA, the environment may consist of a yellow or white page server, mediators, databases or any other entities that serve as productive services in order to support the functionality of the MAS. However, all of these entities are not mandatory in order to represent the environment and can be individually defined for a particular MAS.

Without providing a concrete definition of the actual parts of the environment, an approach that allows for a high-level description or classification of environments is presented in [108]. According to the authors, the environment of a MAS can have different properties which characterize it. They define the following properties an environment may have:

- it may be *fully observable* by an agent if its sensors can perceive the complete state of the environment at each point in time. Otherwise, it is only *partially observable* if the agent cannot perceive the complete state at each point in time or even *unobservable* if the agent has no sensors.
- it may be *deterministic* or *non-deterministic* depending on the level of uncertainty of the result of an action of an agent applied to the environment. If there is no uncertainty about the result of the agent's action to the environment, it is deterministic and non-deterministic otherwise.
- it may be *episodic* if the experience of the agent is divided into atomic episodes which do not affect each other. In contrast, the environment is *sequential* if a decision of an agent may affect future decisions.

– it may be *dynamic* or *static* which means the environment can change while the agent updates itself or it cannot change. This characterization is coupled with the notion of time of the MAS itself.
– it may be *discrete* or *continuous* depending on the different states the environment can reach. If the state set of the environment contains a finite number of states, it is discrete and continuous otherwise.
– it may be *known* or *unknown* which refers to the agent's state of knowledge about the "laws" of the environment. This means that in a known environment, outcomes for all actions are given, while in an unknown one, there may be an unknown outcome for one or more actions.
– it may be *single agent* or *multi-agent* environment from an agent's perspective. If the MAS only contains two different agents, it is a single environment from each agent's perspective. If there are at least three agents in the system and they can all perceive each other, it is a multi-agent environment.

Within this characterization, several properties might be valid for a single environment or MAS respectively.

Interaction

The third part of a MAS is the way the agents interact with each other and organize themselves. As agents, by definition, are independent and autonomous, they can only communicate with each other by sending and receiving messages. Hence, it is crucial to carefully define the interaction protocols and message handling for every agent in a MAS in order to create and maintain a stable system. These protocols comprise all involved agents in a particular communication act, the number and order of messages exchanged, and the maximum negotiation steps. By defining them, an organizational structure or hierarchy is implicitly created, which supports a structured design of a MAS by restricting and ordering agent interactions. [8] The second crucial part within communication and interaction is the way agents handle incoming and outgoing messages. For instance, an agent might only know a few different kinds of messages as its ontology is limited to a specific domain. Then, it is necessary to define how this particular agent handles a message that is not part of its ontology. Message passing further is typically unpredictable because the order of messages might be nondeterministic, and an agent needs to handle various unpredictable circumstances and message orders. [115] provides several basic protocols that can serve as a blueprint for the design of individual ones.

3.3.2 Agent-based Simulation Models

The concept of agents and MAS can either be used to describe an existing complex system or to build an artificial complex system in order to solve a specific task. In both cases, a model of the system is required, which constitutes all required ingredients of a MAS. This model, in turn, can then be used to either simulate and study the observed system (see also Section 3.1.3) or to implement the model designed as MAS in a field set up to solve a specific problem.

The presented theoretical foundations can now be used to distinguish different forms of a MAS based on the constitution of the agents and the environment in which they exist and interact. Figure 3.7 illustrates the now following distinction.

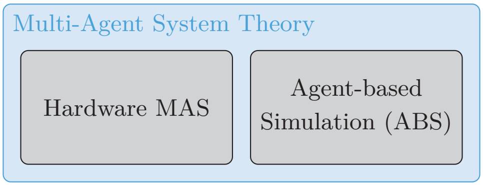


Figure 3.7: Distinction of MAS agent and environment constitution

Although agents are always referred to as software agents, this description is slightly imprecise. Of course, from today's perspective, in most cases, an agent is a software entity in the sense that its behavior is defined in terms of software code. However, depending on the actual specification and purpose of the MAS this piece of software may interact and influence either a real-world environment or a simulated environment. Following these considerations, MAS can be differentiated further either as hardware or real-world MAS or pure software or simulated MAS:

Hardware or Real-World MAS In this kind of a MAS, agents exist not only as software but also as hardware entities. They are located either in a hardware testbed or in the field and their decisions and actions may influence the real world. Hence, the whole environment is not modeled and simulated but is a real-world environment. This view brings MAS close to the field of autonomous components as presented in [116] and makes them attractive for the application in power systems. For example, [117], [118] apply them in the context of distributed control, [119], [120] use them for microgrid control applications and [121] reviews their application for power system protection.

Purely Software MAS A pure software MAS is the simulation and hence execution of a simulation model, whose basic structure is a MAS. Other terms for this kind of MAS are multi-agent system simulation, agent-based modeling and simulation or just ABS. More concrete, this means that the main active entities and components can be identified as agents and their implementation is based on agent-related concepts and technologies, e.g., message passing, encapsulated data structures and interaction protocols. ABS allows the observations of the model under investigation of at least the agent and the global system level. This property is in particular interesting in domains where emergent behavior like biology, social science or traffic on a global scale may occur. However, emergent behavior is not a requirement in order to gain a benefit from ABS. Its modularity also makes it a viable approach and paradigm to develop complex simulation systems where the relation and interaction between different components matters. [8]

Since this thesis develops an agent-based simulation model, ABS and its components are of particular interest. Therefore, in addition to the general aspects of MASs described in Section 3.3.1, more information about the ingredients of ABS are given in the following and for the remainder of this thesis, only ABS are considered.

According to [8] a multi-agent simulation model constitutes of an agent system model, an environment model as well as of simulation infrastructure, as already illustrated in Figure 1.2. Each of these ingredients has the following properties:

Agent System Model The agent system model consists of all simulated agent models that exist within the simulated environment model. Each agent is autonomous with respect to any other agent and the environment. However, there is no autonomy in relation to the modeler who designed the agent system model. All agents exist within a simulated but no real-world environment. Their state, internal behavior and goals are described by internal state variables. All interactions between agents need to be defined by formulating interaction and negotiation protocols with respect to the agent's individual state and behavior. Interaction with other agents is only possible via message exchange and all manipulation of the environment may, in general, be perceptible by all other agents in the system.

Environment Model As specified in the general definition of MAS, all agents exist within an environment. In the case of ABS, this means that all agents exist in a modeled and simulated environment. Within this environment, agents may interact with each other and, if the environment is explicitly modeled, with the environment entity itself. As the real world does, the modeled and simulated environment also may constrain possible agent interactions or visibility of certain parts of the environment for specific agent types. In contrast to real-world environments in a hardware MAS, ABSs environments do not have an implicitly naturally given notion of time - neither physical nor logical time. Hence, time and the synchronization of agents and the environment with it needs to be modeled explicitly using time representation techniques, e.g., as presented in Section 3.1.3.

Simulation Infrastructure The simulation infrastructure provides all required components allowing the execution of the agent system model and the environment model. It provides the underlying computational framework that allows for message passing, time advancement, data flow, input and output modules and every other required instrumentation. As [8] points out, the borderline between simulation infrastructure and simulation environment is sometimes fluent and cannot be drawn precisely for every ABS.

3.3.3 Agents, Objects and Actors - Implementing ABS

ABS can be implemented using either generic or domain-driven frameworks or toolkits, Agent-oriented Programming Languages (APLs) or generic programming languages like C++, Python or Java. [122] No matter which of these options is selected, it makes sense to have at least a broad understanding of the relation between MAS theory and existing programming paradigms that allow for the construction of ABS. This understanding may help to implement an ABS from scratch or to apply a selected framework. Additionally, although several frameworks are available, sometimes none exactly match the application case or provide the required flexibility for a specific task. As APL have not been widely adopted so far, the main focus of the following section is on paradigms of generic programming languages. [123] Readers interested in a broad up-to-date overview on APL may refer to [124].

Based on the general properties of agents presented in Section 3.3.1 two programming paradigms can be identified, which share common ideas with MAS and can be used to implement ABS. The first one is Object-oriented Programming (OOP), the second one is the concept of *actors*. The following section outlines the shared ideas between each paradigm and MAS and subsequently discusses their subtle differences. The section is deliberately limited to the shared ideas and no comprehensive overall description of the respective implementation paradigms is given. General overviews are given by [125] for OOP and [126] for actors.

Object-oriented programming

As [105] points out, modelers familiar with the concept of OOP may see similarities between objects and agents. In OOP, the main computational entity is an object which *encapsulates* some state data and is able to perform actions using its *methods*. In fact, agents are sometimes even referred to be a special kind of objects. [127] While at first glance, the concept of objects is very similar to the concept of agents, a detailed examination reveals some essential differences which differentiate them from agents. All differences presented in the following refer to the standard object-oriented programming model and are taken from [105]:

An object's execution of an action can be triggered by calling the corresponding public method and the implementing object has no control over its method's executions. It neither can refuse the execution nor can it control which other entity can execute the method. In contrast, agents do not provide a similar concept as a public method. The execution of an agent's action is only possible on a *request* basis, meaning that if an agent i requests an action from agent j, agent j has complete control and can decide if, when and how it *responds* to i's request. Hence agents have more robust control and notion of autonomy in contrast to objects. The slogan "objects do it for free, agents do it because they want to" [105] nicely summarizes this difference. The second difference between objects and agents is that objects are not concurrent by nature, but a MAS is inherently multithreaded. [105] Lastly, objects are not as flexible as agents when it comes to their actual behavior. While agents incorporate concepts like reactivity, proactivity, sociality and autonomous behavior, these concepts are not part of OOP. [105]

Although these differences between objects and agents exist, it is still possible to implement agents and a MAS using the concept of objects and many existing frameworks make use of OOP accordingly. [105], [109] However, in particular, the fact that the OOP concept does not say anything about concurrency and action request requires much effort when building a MAS or ABS without using a framework.

Actor-based programming

The concept of actors is a theoretical mathematical model of concurrent computation, which has been originally introduced in [126]. The model serves as a basis for several implementations of concurrency in several generic programming languages, e.g., C++ [128], Erlang [129] and Scala [130]. An actor represents the main computational entity that may send and receive messages to and from other actors it knows about. It holds a mailbox represented by a queue containing all messages it receives. An actor may create new actors, determines its behavior based on the messages within its mailbox and may modify its private state data, but it can neither read nor modify the state data of another actor directly. The only way to do this is to request information or action execution from another actor via message passing. Furthermore, there is no shared memory between multiple actors to avoid race conditions which, as the race condition that caused the Northeast American Blackout in 2003 in Akron, Ohio showed, can have catastrophic consequences. [131]

While many properties of actors and agents may seem similar, as [114] points out, there exist subtle but important differences between both concepts:

Identity Both actors and agents share the concept of identity. For both concepts, the identity allows for distinguishing each actor or agent from another actor or agent. The concrete identity of an actor is defined by the address of its mailbox or the actor's name. In contrast, an agent's identity can be defined by a name but also by a set of the agent's characterizing properties defined by its abilities. Furthermore, an agent may actively advertise the services or operations it offers, e.g., in the yellow pages of the agent system.

Autonomy While actors and agents share the concept of autonomy, the main difference between both paradigms is the actual maturity of autonomy. As the actor paradigm has been mainly developed to support concurrent computation, an actor's behavior tends to be guided more strictly by a deterministic, pre-programmed behavior. In contrast, the concept of agents includes several more advanced mechanisms for their behavior like reasoning, learning or planning, which may result in non-deterministic behavior. Furthermore, agent theory contains the concept of an agent's agenda that it tries to pursue, while actors are more seen to react to external stimuli.

Communication Communication in both paradigms is message-based, asynchronous and concurrent. Furthermore, in both concepts, the actual interaction between entities, agents or actors, needs to be well pre-defined in detailed interaction or negotiation protocols. However, one of the small but significant differences between the agent and actor paradigm is how a message recipient is defined. In the actor paradigm, a recipient is identity-based, which means that when a message from one actor to another is sent, the recipient's identity needs to be known and specified and the message's content is only meaningful to its recipient. No other intermediaries will attempt to read or interpret the content of the message. Agent theory, in contrast, also includes the concept of content-based messages. Content-based messages interpretation means that the message's structure and actual values determine the message's recipient. An intermediate entity may read the message, interpret its content and redirect it to a recipient that can process and respond to the message. Hence, actor-based communication seems to be more data-driven, while agent-based communication seems to be driven more by "speech acts" or explicit intentions.

Coordination Again motivated by their background, coordination of actors focus mainly on programming-language issues like synchronization, resource allocation or performance. In contrast, coordination in agent systems concentrates on forming an organizational structure based on the beliefs and plans of agents in order to reach either an individual or a common goal, and the emphasis is more on knowledge sharing and reasoning.

While both concepts have a lot in common, as outlined above, actors still are less complex than agents. This difference originates from the initial application scope actors and agents have been developed for. However, there is a reason why actors are sometimes referred to as a simpler version of agents. Properly implemented and extended, they can be used to simulate more advanced behaviors and hence are suitable to build complex agents and ABS. [109] In particular, when simulation performance matters, actor-based programming has turned out to mostly outperform APLs as shown by [132].

3.3.4 Summary

In the preceding sections, an overview of the fundamentals of MAS and their relationship to ABS is given. While MAS theory is a general concept that can be used for either real-world applications or simulation purposes, ABS is the application of this theory of modeling and computational approaches to build and run simulation models in order to analyze complex systems. In conjunction with the theory, different ways to implement an ABS are shown. A particular emphasis is put on the application of generic programming languages and concepts which can be used to build a flexible, framework-less ABS. It can be concluded that the eventually chosen approach implementing an ABS depends on the specific application case, the knowledge of the modelers involved as well as different boundary conditions like required scalability and model performance. In particular, when using generic programming languages, either an OOP or actor architectures seems to be close to MAS theory and can be a good starting point for their implementation. While actors are conceptually closer to agents and have their origins in the field of scalable, high concurrent computation, they seem to be more suitable in order to build up large-scale ABSs which need to provide good scalability and performance. However, they come with an increased complexity introduced by message passing and concurrency in general. In contrast, if scalability does not matter that much or when rapid prototyping is required, the application of OOP techniques seems to be a reasonable and suitable approach to build an ABS.

4 Agent-based Discrete-Event Electric Power System Model

This chapter introduces SIMONA, a newly developed agent-based discrete-event simulation model. Taking into account the preliminary work of [9] and [10] the existing system is completely redesigned from a methodical and implementation perspective. Some of the previous model's shortcomings are eliminated in this process, and its capabilities are advanced from a pure distribution grid planning approach towards a flexible, holistic simulation model. Grid and system participant time series, generated by the newly developed simulation model, can support active distribution system planning, operation, and analysis.

Particular emphasis has been put on a detailed representation of electric system participants and their interaction, straightforward integration of new physical models, and a flexible and efficient interface that allows for the connection of other model environments or co-simulation frameworks.

The chapter starts with a short model specification in Section 4.1 followed by a general overview of the model and design concept in Section 4.2 and an introduction to the different physical models, including the agent terminology used within SIMONA in Section 4.3. Section 4.4 then introduces the new discrete-event-based time advancement mechanism as well as the model of the responsible scheduling entity. Section 4.5 extensively presents a new multi-voltage level distributed backward-forward sweep power flow algorithm, including the underlying theoretical ideas. Furthermore, the integration of the algorithm in the agent environment, the responsible agent entities, and their interaction protocol are presented. In Section 4.6, a new physical model of a Photovoltaic (PV) plant is developed, combining direct irradiation, diffuse irradiation, and reflected irradiation to calculate power feed-in into one single model. The chapter closes with an evaluation of the developed model regarding the defined specifications.

It is important to mention that the overall model developed is based on the joint research efforts of Chris Kittl and the author. Therefore, this thesis only some selected parts of the overall model, and these parts primarily or exclusively reflect the author's work. More information on the overall model, in particular about system participant interaction, concrete two- and three-winding transformer models, as well as their comparison with other existing models, can be found in [11].

4.1 Overall Model Specifications

Within the analysis stage of the modeling and simulation cycle depicted in Figure 3.2 not only the system to be represented as a model but also additional requirements, like research objective and intended application cases, need to be taken into account. Doing so is the only way to find a good compromise between the detailed representation of the real system and model abstraction. Therefore, the specifications of the developed model must, at least shortly, be described.

As the modeling and simulation cycle is an iterative process, they might be adapted throughout several iterations. Furthermore, they can be used to compare the actual resulting model capabilities with the defined requirements. Additionally, as the actual implementation of the model also impacts its performance and outcome, model and implementation specifications are interconnected. While several iterations have taken place in the course of this thesis, Table 4.1 solely reflects the latest version of the specifications. The same is true for the overall model of SIMONA presented in this thesis. It is the outcome of several iterations and represents the latest, most advanced version of it.

Table 4.1: Simulation model specifications overview

General Model Specification
Description of the notion of an agent
Flexible input and output data time step length
Partial up to full consideration of pre-calculated exogenous data
Controllability by and consideration of data of co-simulations
Standalone system participant simulation
System Participants
Detailed bottom-up models with focus on single entities
Consideration of entity interdependencies
Consideration of individual behavioral aspects
Clear separation between behavioral and phyiscal model
Validity of the physical model
Grid Representation and Power Flow Execution
Validity of the physical model
Exchangeability of the power flow algorithm
Concurrency by design for scalability

Several different factors drive the determination of model requirements and resulting specifications in SIMONA's case. First of all, the anticipated future use of the model, its use for planning, operation, and system analysis purposes, plays a significant role. This broad range of potential application cases requires considering several aspects concerning model complexity, level of simulation result detail, performance, and usability. Furthermore, the target group of the simulation model, which spans users in the field of research and practice, impacts the model design. Finally, the experience and feedback gained from the development and application of the existing simulation model of [9] and [10] is of high value for the redesign. In this sense, the newly developed model introduced in this thesis represents the latest iteration loop within the modeling and simulation cycle.

With the target groups being Distribution Grid Operators (DSOs) and researchers in the field of multi-energy systems at the distribution grid level, the anticipated future use is the generation of distribution system time series. The need for these time series is not limited to grid operating point and utilization but also covers a wide range of system participants. In particular, because of the mutual interdependencies between system participants and the grid, detailed information about the behavior of each system participant is required. Therefore, not only their grid utilization time series should be calculable, but the model should also be able to generate system participant-only time series. Additionally, time series generation should not be limited to the current situation but also allow for the simulation of different anticipated future scenarios of active distribution grids. A potential application case may be the influence of new technologies, entities, or control algorithms on the energy system. The applied agent-based methodology is of particular interest for such cases, as it allows but does not require detailed representations of single entities, their interaction, and individual goals. By simulating a large number of these entities, their local and regional impact on the energy system and potential emergent behavior of the entity collective can be investigated. Extending this application case from one sector, e.g., the electricity grid, to sector-coupled multi-energy system analyses may lead to new insights and a large variety of additional application cases to develop future energy distribution systems.

From a historical perspective, energy system modeling and simulation have been primarily closed and proprietary. [62] An indication of this is provided in particular by the large number of proprietary tools, as presented in Section 2.3. This situation makes it hard to almost impossible to use or validate state-of-the-art models. However, a slow but increasing trend towards more open energy modeling can be observed in recent years. Motivated by the transformation of the energy system, it seems that finally, energy models and their implementation are more often freely available. The concepts and benefits of OpenSource (OS) or at least OpenCore (OC) started to be recognized and accepted by energy researchers and practitioners. Following these developments, another specification regarding the intended target group is that the model developed within this thesis should be publicly available as OS code to be used, validated, and extended by interested researchers and practitioners. This further implicitly induces several specifications in order to support the application of the developed model by others. First of all, a broad range of standard system participant models should be available. Secondly, if a user intends to integrate their own physical or behavioral system participant model, this should be as simple as possible. Thirdly, the underlying algorithm should be interchangeable regarding power flow calculations to allow, e.g., DC power flow calculations only if required. Furthermore, the required electricity grid model should be compatible with existing grid models or, at least, converters should be provided, making existing proprietary and non-proprietary grid data available to the model. Finally, for multi-energy system analysis, it should be possible to provide external data or integrate the developed model into existing co-simulation frameworks. Parts of these requirements do affect not only the model structure but also its concrete implementation.

Reviewing existing work and taking into account feedback and review comments is good practice in the field of research. Therefore, also the feedback[7] to the concept of the existing model presented in [9] and [10] has been considered when formulating the model specifications for the new simulation model. During the application of the existing system, it has turned out that the way it has been modeled and implemented prevents the execution of investigations at large grid structures. The reason for this is, on the one hand, the concrete model structure, on the other hand, its implementation. Therefore, being scalable and applicable to large grid structures becomes a specification for the newly developed model. Further feedback on the existing model has also been that it is not usable and validatable because it has not been published OS. This aspect further justifies the OS specification. Finally, several times the existing tool has been presented in the research community, the applicability of the term *agent* in the context of the developed system has been questioned. As discussed in Section 3.3, the notion of an agent differs from domain to domain and model to model. Therefore, whenever Multi-Agent System (MAS) or Agent-

[7]The developed model has been part of several research projects and publications. Scientific feedback has been provided by reviewers of publications and during discussions in expert groups and at conferences. Feedback from the practice has been gathered within the three research projects *Agent.Netz*, *Agent.GridPlan* and *NOVAgent* in which different DSOs and other companies were involved. Currently ongoing research activities still provide valuable feedback for further model development.

based Model (ABM) is developed, at least having a few remarks on the notion of an agent within the developed model is required. Describing the notion of an agent within the developed model is taken into account as specification accordingly.

The fulfillment of these requirements, summarized in Table 4.1, is the guiding principle in the development of the model presented in the following.

4.2 Conceptual Overview

Modeling the electric power distribution system from a bottom-up perspective using an agent-based approach is straightforward. By nature, the distribution grid consists of several connected voltage levels, e.g., low-, medium- and high-voltage. Depending on the specific voltage level and regional circumstances, the grid consists of different types of grid assets and system participants connected to the grid's nodes. From a grid perspective, these system participants can either consume or feed-in electricity. Considering this characteristic, representing the grid and its participants as a model using agent-based modeling techniques can be carried out by choosing an appropriate level of abstraction based on real-world circumstances.

The following section describes the constituent elements of the overall system and their interplay before diving deeper into selected parts developed in this thesis.

4.2.1 System Concept

Based on the introduced requirements, a new microservice-inspired model architecture including several new models has been developed.[8] While the core parts are still part of the modularized core of SIMONA, in particular, input and output model parts have been outsourced and transferred to encapsulated modules connected via interfaces. The new concept results in a closed, self-contained system already presented in Figure 1.3. Figure 4.1 shows a simplified aggregated representation of SIMONA, which will be used in the following to explain several of its core concepts. The concept provides several different interfaces that allow for a model and simulation control and the feed-in of external data from a variety of sources. Formally, the resulting overall system follows the general definition of an ABM presented in Equation 1.1.

[8] Microservices are small, autonomous services that, combined in a microservice architecture, form an executable application. [133] This architecture fits well with the combination of an agent-based simulation model and actor-based programming. Therefore, to simplify the implementation, a microservice-inspired structure is already adopted in the modeling process. A comprehensive overview of microservice architectures can be found in [133].

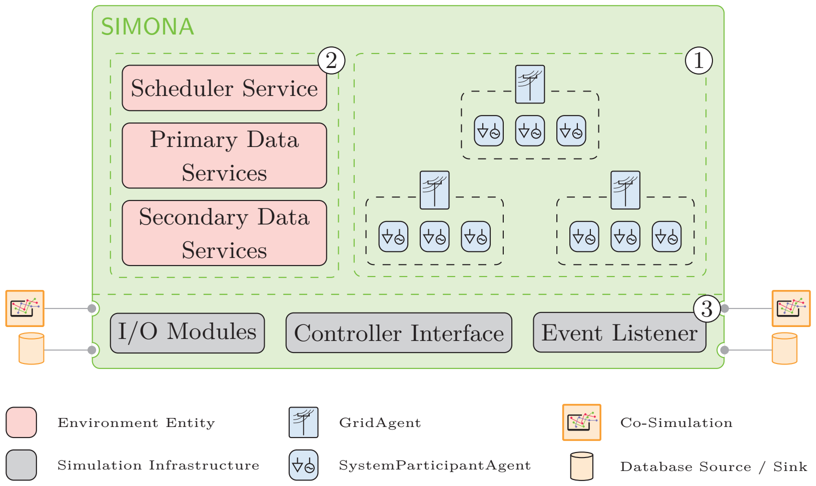


Figure 4.1: Simplified aggregated SIMONA concept overview

Within this system, the following three main parts are available:

① **Agent System Model** This model represents the core and most characterizing part of the overall model. It consists of concrete autonomous agent entity instances interacting with each other.

② **Environment Model** The environment model is the representation of the actual real-world environment. It consists of all models required to provide the artificial environment the agents are existing and interacting in. It includes the resource model RM, the description of global properties GM, the global environment actions function $act_{\text{env}}(\cdot)$ and the execution function $ex(\cdot)$. In SIMONA, these properties and functions are not explicitly modeled as entities but implicitly given through the scheduler, primary and secondary data services. Additionally, the scheduler service represents logical time T which becomes part of the environment model.

③ **Simulation Infrastructure** This part supports model execution. While it is not part of the model, nor does it influence the model's calculation outcome, it is required in order to execute the Agent System Model and the Environment Model in the context of a simulation run.

4.2.2 Agent System Model

The Agent System Model is the main component of the model. In its current state, it consists of several instances of two different agent entities, *GridAgents* (*GAs*) and *SystemParticipantAgents* (*SPAs*). Each *GA* is responsible for a galvanically isolated subgrid, including the system participants connected to it. Responsible in this case means that it represents the physical properties of the underlying electricity grid and coordinates the interaction between the electricity grid and the connected system participants. Each system participant, e.g., a Wind Energy Converter (WEC) or PV unit connected to the grid, is represented by a *SPA* correspondingly. The actual agent type of these two entities, which determines their functionality, may vary depending on their task. In particular, the specific characteristic and shape of a *SPA* highly depend on the physical model it contains. For a detailed description of the agent types available in SIMONA, as well as an overview about the different system participants a *SPA* can represent see Section 4.3. During model execution, the individual agents interact with each other, generally resulting in the abstract communication flows depicted in Figure 4.2.

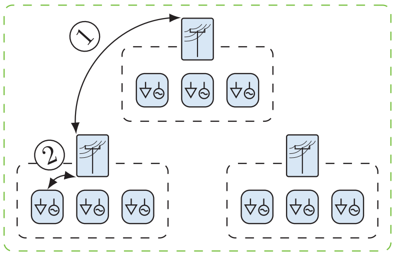


Figure 4.2: Agent System Model interaction

The agent communication protocols are designed such that each *GA* takes note of its superordinate and subordinate *GAs* and all *SPAs* physically connected to its subgrid. Each *SPA* can handle any communication request by every *GA*. Yet, the fact that currently each *GA* only knows about the *SPAs* in its subgrid creates a de facto stable communication channel between one *GA* and its *SPAs*. This approach provides the flexibility to extend the model's capabilities when new interactions, e.g., arising from a new smart grid, smart market or energy management applications, should be investigated. In such a situation, the existing communication protocols can is easily adaptable to enable additional interactions, e.g., between multiple *SPAs*.

The *GAs* main task is the determination of power flows in its subgrid. To achieve this, it has to communicate with the superordinate and subordinate *GAs* ① and, in addition, it has to request the current feed-in or consumed power of *SPAs* connected to its grid ②. In turn, the task of the *SPAs* is to determine its power feed-in or consumption and to answer the power request queries of their *GA*.

Compared to the abstract, general representation of communication in Figure 4.2, the concrete communication protocols and algorithms are significantly more complex. For this reason, they are explained in detail in their subchapters. An in-depth explanation why each *GA* is responsible for a single, galvanically isolated subgrid, its intern functionality as well as the underlying Distributed Backward-Forward Sweep Algorithm (DBFS) is given in Section 4.5. For the general functionality of the interaction protocols of *SPAs*, please refer to the description of the generic system agent in [11]. Furthermore, a concrete example of a *SPA* in the form of a *PvAgent* is presented in Section 4.6.

4.2.3 Environment Model

The primary purpose of the Environment Model is to provide an artificial representation of the real world the agents exist in. Consequently, it provides all required global state information, e.g., current logical time or synchronization protocols, to the Agent System Model. Inspired by a microservice architecture, the Environment Model consists of different services that are largely decoupled from each other. It is composed of three main service types, *Scheduler Services*, *Primary Data Services* (*PDSs*) and *Secondary Data Services* (*SDSs*). Each of them is shortly introduced in the following.

Scheduler Services

The task of a scheduler service within an agent-based discrete-event simulation model is to manage the chronological correct execution order of events and correspondingly implicitly to provide the current logical time to the involved entities. Only that way, a stable logical simulation time advancement can be guaranteed. Therefore, maintaining at least one list of simulation events is mandatory to execute the overall model. This task, the management of one or several event lists, is carried out by *Scheduler Services*. Hence, all scheduler service instances can be considered the single point of truth regarding simulation or logical time t. As part of the Environment Model, they consist of at least one *SimScheduler* (*SimS*) entity. Each

SimS contains several priority-ordered queues containing different types of events, sorted by their scheduled time. Regardless of the order in which events are added to the queues, they are sorted, handled, and removed in strictly chronological logical time order. If there are several *SimSs* instances available, each entity does not only contain event lists but also functionalities that allow for a synchronization of all scheduler service entities. The concept of having multiple *SimSs* available can be helpful when a large number of event-emitting entities is simulated because it allows the distribution of event handling workload on multiple, concurrently acting *SimSs*. As the scheduler entity represents a core component of SIMONA, Section 4.4.2 describes its functionality in detail.

Data Concepts & Flows in SIMONA

In order to allow flexible use of SIMONA for different use cases, it is possible to connect different data sources and sinks to its interfaces. As this connectivity is closely related to the Environment Model, the following section outlines the data concept and the data flow of SIMONA to facilitate understanding of the subsequently following *PDSs* and *SDSs* descriptions.

SIMONA interfaces distinguish between *primary* and *secondary* data, whereas both data types can be generated externally or internally. The resulting external input-output-data concept is depicted in Figure 4.3. The magenta arrows depict primary data, and the blue arrows depict secondary data.

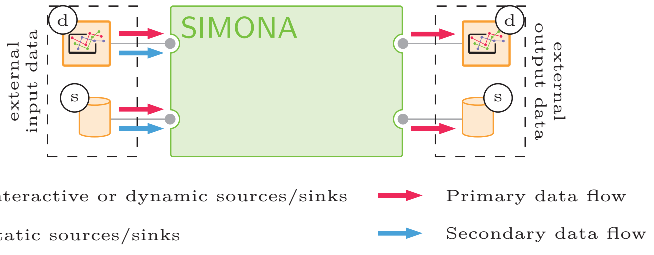


Figure 4.3: Data concept and flow of externally provided data in SIMONA

No matter if internal or external, primary data is everything that can be considered to be a technical result of a physical model calculation, e.g., apparent power of a WEC unit model or the actual utilization of a line model. It is relevant for the results of the model and may be required for further analysis. Additionally, the primary data of one agent can become secondary data for another agent. For example, if a PV unit calculates its power in-feed, the resulting apparent power is secondary data from the grid perspective, as the grid requires the power in-feed in order to execute power flow calculations and to derive line or transformer utilization. Consequently, all data used to calculate primary data are called secondary data. For instance, wind velocity can be classified as secondary data from a WEC unit perspective, as it is used in order to calculate the actual, apparent power as primary data.

As already stated, primary and secondary data can be provided and consumed externally and internally. External input data is coming from sources outside the scope of the Environment Model. This means the data is not part of the SIMONA model itself but provided from either static ⓢ or interactive dynamic ⓓ external sources. For example, a database is a static external source or sink and a co-simulation is a dynamic external source or sink. External data is always provided from the Environment Model to the Agent Model. The underlying capabilities required to enable the Environment Model to access external data are provided by the Simulation Infrastructure. This holds true for external input data as well as for external output data. In contrast to external input data, the external output is always the result, and hence, primary data. Like external input data, external output data is always provided by the Environment Model making use of the Simulation Infrastructure.

Primary Data Services

PDSs are entities representing the artificial model of entities within an external environment that is not part of the SIMONA Environment Model but whose primary data values should be taken into account by the Agent System Model. Every single *PDS* is responsible for the provision of one type of externally provided primary input data. The sources of this data might vary from measurement data, data from any co- or Hardware-in-the-Loop-simulation, or any other pre-calculated static data from, e.g., an external database. In order to provide externally available data to the Agent System Model, it requires and hence is strongly coupled with the Simulation Infrastructure component. On a high level, *PDSs* can be pictured as the primary interfaces that enable SIMONA to be connected to data sources outside of its main scope. An application case could be, for example, the consideration of field measurement data for active and reactive power of individual PV units in a grid to be investigated. In this case, for each unit, a *PDS* would exist that provides the corresponding digital representation of the PV unit *SPA* with the measured data. The

SPA would then consider the measured data available instead of its virtual model. All other *SPA* for which no measured data are available would use their digital system participant models accordingly. The resulting hybrid simulation would allow for power flow calculations in the *GA* taking into account measured and simulated data of the system participants. This kind of simulation could be used for forecasting purposes in a grid control room.

Formally, a single *PDS* can be represented by a time dependant function $f : \mathbf{T} \rightarrow \mathbf{D}$ providing a n-tuple of external primary data values d for each point in logical time t to specific *SPAs*. The resulting function can be written as $f(t) = (d_1, d_2, d_3, ..., d_n)$. It supports the provision of heat, active power and reactive power values.

Secondary Data Services

SDSs are entities that represent the artificial model of any external environment that is not part of the SIMONA Environment Model but whose secondary data values should be taken into account or are even required by agents within the Agent System Model. At the moment, *SDSs* comprise three different concrete service entities, a *WeatherService*, a *MarketService* and an *EvService* as shown in Figure 4.4.

Secondary Data Services
WeatherService
MarketService
EvService

Figure 4.4: Secondary data services in SIMONA

Exactly like all *PDSs*, all *SDSs* are closely linked to the Simulation Infrastructure. Thus, the origin of the provided data is initially irrelevant for the services, as long as it matches the expected data format. The format, as well as the expected values, differ from *SDS* to *SDS*. For example, it would be possible in principle to make real-time measurement data from a weather station accessible to the Agent System Model using the *WeatherService*. However, the service also works with historical data provided from a static database or similar.

WeatherService
The *WeatherService* consists of one or more entities that provide all requesting agent entities with weather data. In most cases, the requesting agent entities are *SPAs*. Nevertheless, also *GA* weather requests are possible, e.g., if a subgrid contains lines supporting overhead line monitoring, which might require tempera-

ture or wind velocity information. More formally, the general functionality of the *WeatherService* can be described as a function $f : \mathbf{T} \times \mathbf{G} \rightarrow \mathbf{W}$ which provides an n-tuple of weather data values w for each point in logical time t to the requesting agent with the geographical coordinates g represented by its latitude and longitude coordinates. The resulting function is $f(t, g) = (w_{1,g}, w_{2,g}, w_{3,g}, ..., w_{n,g})$. The available set of different types of weather data values in $\mathbf{W}$ depends on the configured underlying weather data model. Depending on the underlying weather data model and the requesting agent, weather data may not be available for the agent's exact geographical position g. In this case, the *WeatherService* determines the resulting weather data for a given coordinate as a linear geographical-distance-weighted approximation.

The approximation of weather data starts by selecting m different coordinates for which weather data is available and which are the closest coordinates to the requesting agent's geographical position. Afterward, the distances $d_{r,i}$ between the requesting coordinate r and each of the m coordinates are calculated using the haversine formula Equation 4.1–4.3, where ϕ is the latitude and λ is the longitude of a coordinate, R_{E} is the radius of the earth and i is the selected coordinate.

$$d_{r,i} = R_{\mathrm{E}} \cdot c_{r,i} \tag{4.1}$$

$$c_{r,i} = 2 \cdot \operatorname{arctan2}\left(\sqrt{a_{r,i}}, \sqrt{1 - a_{r,i}}\right) \tag{4.2}$$

$$a_{r,i} = \sin^2 \frac{\phi_i - \phi_r}{2} + \cos \phi_r + \cos \phi_i + \sin^2 \frac{\lambda_i - \lambda_r}{2} \tag{4.3}$$

The linear and reciprocal weighting factor for each of the m selected coordinates is determined using the resulting distances in Equation 4.4.

$$\kappa_i = 1 - \frac{\frac{d_{r,i}}{\sum_{i=1}^{m} d_{r,i}}}{m - 1} \tag{4.4}$$

Finally, the weather data that is provided to the requesting agent is determined by summing up each of the n weighted weather data values resulting from the weather service function $f(t)$ for each of the m coordinates using Equation 4.5. The result represents an acceptable approximation of the weather data available at the geographical position of the requesting agent and is provided accordingly.

$$w_{n,g} = \sum_{i=1}^{m} \kappa_i \cdot w_{n,i} \tag{4.5}$$

MarketService
The *MarketService* consists of one or more entities that provide all requesting agent entities with market price data. The wholesale electricity market price is currently supported, which is normally provided by historical market price time series. However, it is also possible to couple the model environment with an external co-simulation that provides simulated market price time series. Formally speaking, the *MarketService* can be described as a function $f : \mathbf{T} \rightarrow \mathbf{P}$ which provides a price data value p for each point in logical time t to the requesting agent. The resulting function can then be expressed as $f(t) = p$. In order to allow for the consideration of different price zones when, e.g., smart market mechanisms are investigated, the function can be expanded to $f(t, z) = p_z$ where $z \in \mathbf{Z}$ represents the price zone the requesting agent lives in.

EvService
The *EvService* is represented by one or more entities whose main task is to provide *SPAs* holding a physical electric vehicle charging station model with information about currently connected Electric Vehicles (EVs). EVs are in general not part of the electricity grid. However, EV charging stations are the main coupling point between the electricity grid and EVs. Therefore, *EvService* entities represent the interface to couple another external model of electric vehicle movements with SIMONA to enable the execution of sector coupled simulations and investigations. The complexity of this external EV model may vary from a straightforward Markov-chain-based trip model, e.g., [134], to a fully meso- or even microsimulation, e.g., [135] or [136]. As depicted in Figure 1.3, SIMONA also provides an optional electric vehicle mobility simulation module. The only requirement is the provision of the corresponding data in a specified interface format. Formally, the *EvService* can be described as a function $f : \mathbf{T} \rightarrow \mathbf{E}$ which provides a n-tuple e of EV data values $f(t) = (e_1, e_2, e_3, ..., e_n)$ for each point in logical time t to all *SPA* containing a charging station model. The tuple may consist of all data required by a charging station in order to serve the EV that should be charged considering the circumstances of the electricity grid. This required data depends on the actual case of investigation and may include arrival and departure time, EV State of Charge (SoC) and acceptable electricity prices.

4.2.4 Summary

This section presents several core concepts of the developed agent-based simulation model from a high-level perspective. This overview includes the general system concept, depicted in Figure 4.1, with the **Agent System Model**, the **Environment Model** and the **Simulation Infrastructure**. Additionally, the main ingredients of each of these models are introduced.

4.3 Agent Classification

The loose and not very specific definition of an *agent* is both an advantage and a disadvantage. The advantage is that the term applies to a variety of different entities in a model. The disadvantage, however, is precisely this broad applicability since it is no longer clear what distinguishes an agent from other entities such as environmental services belonging to the model environment.

In SIMONA, all relevant entities are modeled as agents. In the overall model, these agents are exclusively technical agents, representing a technical system. An agent can be either, e.g., a *SPA* in the form of a PV or WEC unit or even a representation of an electricity grid. Accordingly, each agent contains a physical model of the actual asset or system it represents. The focus of each agent is, on their technical result data, meaning that specific technical calculations are of main interest. Examples for such calculations are feed-in or consumption behavior of system participants or electricity grid utilization, e.g., line or transformer utilization. Therefore, in order to provide an agent description, first of all, it is necessary to outline the common understanding of physical models in SIMONA.

4.3.1 Physical Models

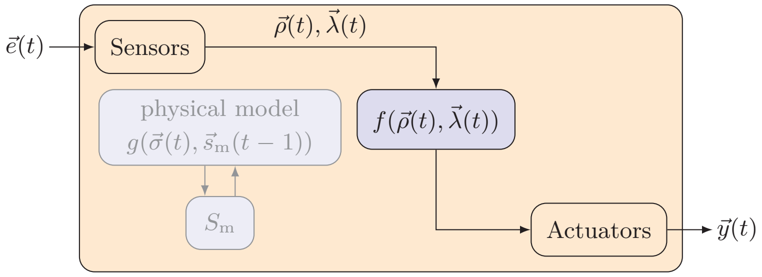
Focusing primarily on the energy system's electrotechnical part, the current model only contains technical agents. Nevertheless, an extension by partly technical or non-technical agents is possible. At the moment, however, each agent holds a physical model of the respective technical unit it represents. A physical model is the mathematical representation of the actual system it represents, transforming secondary data received by the agent to primary data. Depending on the specific physical model, the temporal sequence of operations of the model can be relevant or irrelevant. In the first case, it is a stateful model and in the second, a stateless model. An example of a stateful physical model is an electric battery. In the context of a simulation, the SoC at time t influences the model's physical capabilities and depends on the operation and the resulting state of the model at time $t-1$. In contrast, there is, for example, a WEC, whose primary data provision at any time t depends only on the secondary data available at the corresponding time.

4.3.2 Agent Types

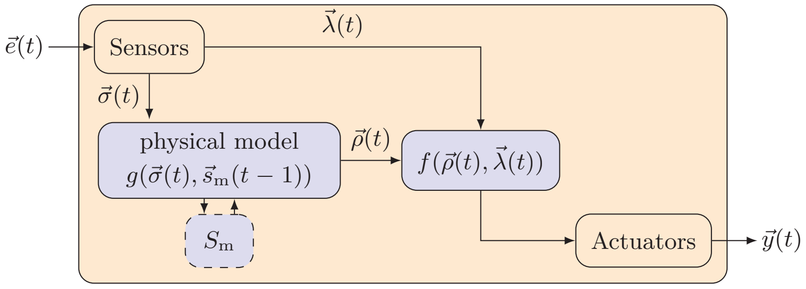
The specifications significantly determine the characteristics of the individual agent types for the developed overall model. Three high-level agent types exist in SIMONA, whereby each type defines the agent's capabilities in terms of its interaction with other agents and the environment and its non-technical decision behavior. In general, all agent types are polymorphic, which means the type of an agent can change from one application case to another. Hence, the concrete agent type is determined by the specific application case. For example, for a particular application case, it might be necessary to provide external primary PV data, while in other cases, a physical model is required. While in the first case, a Type 1 Agent is sufficient, the second case requires a Type 2 Agent with a concrete physical model. However, both types can be represented by the same polymorph model entity.

All agents consist of the general model parts required to interact with their surrounding environment and a specific physical model part. This differentiation between the agent requirements like perceiving and reacting on its environment and the physical model part provides the required high degree of overall model flexibility.

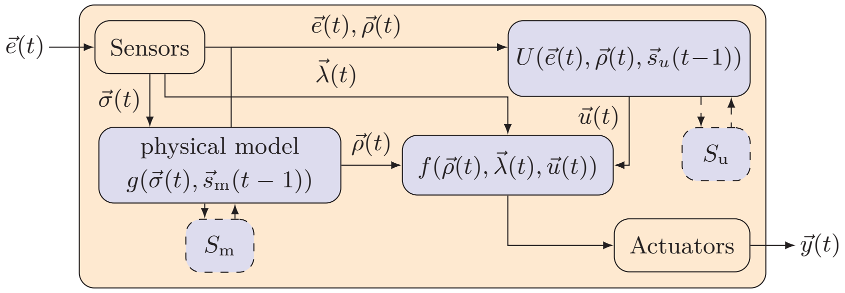
All agent types can perceive the state of their environment $\vec{e}$ at time t using their sensors and act on it by creating an action $y(t)$ using its actuators. The perceived surrounding environment $\vec{e}(t)$ is a tuple of either primary input data information $\vec{\rho}(t)$ or secondary data information $\vec{\sigma}(t)$, either from external data services or provided from another agent, and general communication requests $\vec{\lambda}(t)$. All information preceived or send by the agent depend on the current point in logical time t. Furthermore, all agents contain a state transition or next action decision function $f(\cdot)$. Depending on the agent's type, this next action function considers different input parameters to determine the next action of the agent. Figure 4.5 depicts the different agent types with their internal structure.


$\vec{e}(t)$
Sensors
$\vec{\rho}(t), \vec{\lambda}(t)$
physical model
$g(\vec{\sigma}(t), \vec{s}_{\mathrm{m}}(t-1))$
$f(\vec{\rho}(t), \vec{\lambda}(t))$
S_{m}
Actuators
$\vec{y}(t)$

(a) Type 1 Agent: Agent model with a simple decision function and externally provided primary data proxy capabilities



(b) Type 2 Agent: Agent consisting of a decision function, a physical model and an optional physical model state value store


$\vec{e}(t)$
Sensors
$\vec{e}(t), \vec{\rho}(t)$
$U(\vec{e}(t), \vec{\rho}(t), \vec{s}_u(t-1))$
$\vec{\lambda}(t)$
$\vec{\sigma}(t)$
$\vec{u}(t)$
physical model
$g(\vec{\sigma}(t), \vec{s}_{\mathrm{m}}(t-1))$
$\vec{\rho}(t)$
$f(\vec{\rho}(t), \vec{\lambda}(t), \vec{u}(t))$
S_{u}
S_{m}
Actuators
$\vec{y}(t)$

(c) Type 3 Agent: Agent with a decision function, a physical model, an utility function and optional physical model and utility state value stores

Figure 4.5: Agent types available in SIMONA

Type 1 Agent / Simple Proxy Agent Compared to the other agent type, a Type 1 Agent represents the least developed agent type available. The agent only consists of a general agent hull required to communicate with the environment and other agents. Although it holds a physical model, this model is not activated in order to handle secondary data. Instead, the agent uses a short circuit to serve externally provided input data as primary data. Hence, it acts as a proxy for the provided primary data and handles them as if they are a result of its model calculation to answer other agents' requests. To achieve this, the agent relies on predefined primary data values $\vec{\rho}(t)$ for each point in time t. Correspondingly, an externally provided dataset must consists of tuples $\langle\vec{\bar{e}}(t), \vec{\rho}(t)\rangle$ where either a partial representation of the environment $\vec{\bar{e}}(t)$ or the whole environment vector $\vec{e}(t)$ is mapped to one or more specific data points $\vec{\rho}(t)$. In the latter case $\vec{\bar{e}}(t) = \vec{e}(t)$ holds true. An example for such a Type 1 Agent could be a PV agent that holds precalculated power feed-in values mapped to specific points of time, where the time represents the partial representation of the environment.

Type 2 Agent / Physical Model Agent A Type 2 Agent can be seen as an extended version of the Type 1 Agent. The physical model that has been deactivated in the Type 1 Agent is now activated and used in order to heuristically calculate result data values $\vec{\rho}$. To do so, the agent relies on secondary $\vec{\sigma}(t)$ which it perceives from environment $\vec{e}(t)$ for the current point in time t using its sensors. This secondary data is then fed into the physical model which consists of a transformation function $g(\vec{\sigma}(t))$. If the physical model is a stateful model, e.g., an electricity storage system, the agent may also hold a physical model state value store S_m. The physical model transformation function becomes $g(\vec{\sigma}(t), \vec{s}_m(t-1))$ where $\vec{s}_m(t-1)$ is the state of the physical model in the previous point in time.

Type 3 Agent / Physical Model Agent with Utility Function This agent type, as the name already suggests, is an extended version of the Physical Model Agent. It is the most advanced and complex agent type available in SIMONA. The input data the agent is perceiving is again the actual environment state tuple $\vec{e}(t) = \langle\vec{\sigma}(t), \vec{\lambda}(t)\rangle$. Internally, this tuple is split up into its components and distributed to different logic blocks of the agent. The secondary data $\vec{\sigma}(t)$ is passed to the physical model, which calculates the physical model result data $\vec{\rho}(t)$ using the same transformation function as a Type 2 Agent - either with or without a physical model state value store S_m. The result is then passed over to the actual next action decision function f and the utility function logic block. This block does not only receive the physical model result $\vec{\rho}(t)$ but also all other elements of the environment $\vec{e}(t)$. Using the utility function $U(\vec{e}(t), \vec{\rho}(t))$, the agent calculates the current utility $\vec{u}(t)$. This utility function vector might not only contain one specific utility, but several ones for different scenarios or input values $\vec{e}(t), \vec{\rho}(t)$. As with the physical model, depending on the specific use case and the complexity of the utility function, a value store for historically calculated utilities S_u is required. A sample application case could be an agent that uses machine learning techniques and hence requires access to historically calculated values. The resulting utility $\vec{u}(t)$ is then passed over to the extended next action decision function $f(\vec{\rho}(t), \vec{\lambda}(t), \vec{u}(t))$ in order to determine $\vec{y}(t)$.

4.3.3 Agent Entities

The idea of a modular structure that is as flexible as possible does not only has a central role in SIMONA at the level of the overall architecture, but the specific design of the currently available agents is also modularly conceived. Based on the preceding explanations concerning the different types of agents, two concrete agent entity models are available in the Agent System Model. These are the *GA*, representing a single subgrid, and the generic agent model for system participants connected to the grid, the *SPA*. Both of them have already been introduced shortly in the general model overview. A more detailed overview of the agent entities' characteristics and the required data input during overall model execution is provided in the following.

Because the behavior model of each agent entity depends on the specific use case and requires extensive explanations since it comprises the entire behavior and communication properties of the agent, this overview focuses on the physical model and configuration parameters of agents. In particular, while the modular agent architecture allows comparably simple interchangeability of the behavioral model, this simplicity comes with the drawback of an increased complexity of the actual behavior model itself. For this reason, the behavioral model of the *GA* is described in appropriate detail in Section 4.5. [11] provides a detailed description of the generic behavioral model of the *SPA*, including its interaction with service entities. Furthermore, Section 4.6 presents a specific *SPA*, the newly developed the *PvAgent*.

GridAgent

The *GA* represents one of the two central agents in SIMONA and it basically corresponds to either a Type 2 Agent or a Type 3 Agent. It is responsible for determining the utilization of the electrical grid using power flow calculations. The data required to use the *GA* model can be distinguished between initialization data and execution data. In order to be initialized, the *GA* requires the nominal apparent power $S_{\mathrm{n},g}$, the nominal voltage $V_{\mathrm{n},g}$ of its subgrid as well as a physical representation of the grid model the *GA* is responsible for. This model comprises all primary grid assets, consisting of nodes, lines, switches and transformers, and their connection properties that form the grid topology.

In order to use its physical model during overall model execution, the *GA* requires additional data. Following the data flow concept of SIMONA, this so-called execution data is secondary data from the agent's perspective. In the concrete case, this secondary data corresponds to the individual load and feed-in of all *SPA*, which are connected in the corresponding subgrid area, as well as feed-in provided or load requested from all adjacent downstream grid partitions. Formally, the required secondary data of the *GA* can be written as the active power p_n and reactive power q_n of all nodes $n \in \mathbf{N}_g$ in subgrid g:

$$p_n, q_n \ \forall \ n \in \mathbf{N}_g \tag{4.6}$$

Combining this information with the physical model enables the *GA* to execute its next action decision function $f(\cdot)$. In the case of the *GA*, the next action is determined by the actual result of the power flow calculations. The resulting primary data of the *GA* that is used for further calculations and the next action decision are the nodal voltages and currents for all nodes in the grid. Formally, the primary result data of the agents physical model calculations can be written as the complex voltage magnitude $\underline{v}_n$ and complex current $\underline{i}_n$ of all nodes $n \in \mathbf{N}_g$ in subgrid g:

$$\underline{v}_n, \underline{i}_n \ \forall \ n \in \mathbf{N}_g \tag{4.7}$$

SystemParticipantAgent

The second most important entity in SIMONA are the *SPAs*. As already pointed out, this agent entity is formulated generically and its concrete shape in a simulation is determined by its physical model. A *SPA* can have any of the described agent types and like the *GA*, the data required by this agent entity can be distinguished between the actual initialization data and the data relevant for model calculations. The initial data of this agent consists of its physical model and information about possibly required service providers.

However, the execution data is determined depending on the physical model of the agent and can range from none or one parameter to a variety of parameters. Following the data flow concept of SIMONA, the fact that secondary data is consumed by the agent and primary data is provided from the agent is true also for a *SPA*. Table 4.2 shows the different shapes defining physical models that can be used within SIMONA. For an in-depth description of each model and its internal functionality, please refer to the provided references.

4.3.4 Agent States

Agents in SIMONA consist of a behavioral model which is entangled with the general agent hull. In addition, all agents need to reflect the basic structure of a component according to the Discrete-Event System Specification (DEVS) and Equation 3.1. Fulfilling this requirement is crucial to integrate and timely synchronize all agent entities using the modeled time advancement mechanism. The resulting behavioral model can be described using the Finite-State Machine (FSM) formalism. Although this formalism already contains states, these states are equal to component states in the sense of a DEVS but do barely reflect the state of a physical model inside an agent. Therefore, a corresponding agent state is introduced depicted in Figure 4.6.

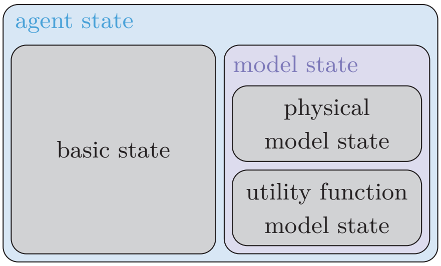


Figure 4.6: Interrelationships between agent and model states

[9]The model presented in [10] has been expanded and revised as part of currently unpublished research activity.

Table 4.2: Physical models available for *SystemParticipantAgents* in SIMONA

Physical Model	**Secondary Data**	**Primary Data**	**Stateful**	**Description**
Fixed Feed-In	–	S	✗	–
Private Household	–	S	✗	[9]
WEC	v_w, ϑ, p	S	✗	[10][9]
PV	t, $E_{\mathrm{e,beam,h}}$, $E_{\mathrm{e,diff,h}}$	S	✗	Section 4.6
BEV Charging Station	t, $\mathbb{E}$	S	✗	–
Electric Storage	S	S, σ_{SoC}	✓	[9]
Heatpump	t, ϑ	S, $\dot{Q}$, $\sigma_{\mathrm{SoC,th}}$	✓	[10]
Biomass Power Plant	t, ϑ, c_{market}	S, $\dot{Q}$, $\sigma_{\mathrm{SoC,th}}$	✓	[B7]
CHP	t, ϑ	S, $\dot{Q}$, $\sigma_{\mathrm{SoC,th}}$	✓	[B5]
Cold Storage	ϑ, $E_{\mathrm{beam,h}}$, $E_{\mathrm{dif,h}}$, c_{market}	S, c_{con}, $\dot{Q}$, ϑ_{in}, $\dot{m}_{\mathrm{goods}}$	✓	[B14]

The target of this agent state is to capture all different characteristics of an agent, which determine its current constitution. Therefore, the agent state is a composition based on a component state extended by the physical and behavioral model state. It fulfills the requirements resulting from the complexity of an agent entity and can be used in order to describe an agent within a FSM formalism. Figure 4.6 depicts the relationship between the agent state, the different model states and the component state or basic state, respectively. Formally, the agent state is a composition of the component state $s \in \mathbf{S}$, called *basic state*, and a newly introduced *model state* $m \in \mathbf{M}$ which consists of a *physical model state* and an optional *utility model state*. While the basic states consist of several general states common to all agents like startup and termination, each agent may define some additional individual basic states. Combining the basic state set with common and individual states and the model state set fully defines the agent state.

4.3.5 Summary

In this section, several core components of the agent model are presented in detail. Particular emphasis is put on the concrete notion of agents, their different types, as well as the agent entities available in SIMONA. Furthermore, the idea of an agent state is presented, allowing for the modeling of agents using FSM and Discrete-Event Modeling and Simulation (DES) theory. The following section introduces the developed logical time synchronization mechanism and the environment model entity which is responsible for time synchronization in the agent model.

4.4 Discrete-Event System Model

As depicted in Equation 1.1, an ABM consists of three parts: a representation of logical time T, a model for the environment $ENVM$ and a model of the multi-agent system AM. SIMONA considers the logical time T as part of the environment model. This section describes the representation of T, the time advancement mechanism, the entity responsible for time advancement in SIMONA, the *SimScheduler* as well as the actual model execution algorithm.

4.4.1 Formal Overall System Concept

Logical time T and time advancement in SIMONA is modeled using a discrete-event approach. More precisely, the overall model is designed as a Discrete-Event Specified

Network (DEVN). There are several reasons this approach is selected to represent logical time in the ABM. First of all, as introduced in Section 3.2, the approach incorporates a modular structure and components have to interact with each other by exchanging messages. This structure and interaction mechanism are close to equal to how agents and their interaction are modeled, hence simplifying building the connection between logical time representation and the agent model AM. Furthermore, it allows the integration of new agents into the overall model seamlessly while allowing proper control to avoid potential deadlocks during model execution. Secondly, from a performance perspective, during model execution, modular coupled models best fit the needs of distributed simulation since their components, agents in this case can be straightforwardly assigned to distributed computational network nodes. [85] Finally, existing implementations, e.g., [137], indicate that using a discrete-event over a fixed-increment time advance mechanism as presented in [9], may improve the performance and execution time of the overall model because it avoids unnecessary model executions.

While the main formalism has already been introduced in Section 3.2.2 using Equation 3.3-3.5 some additional formal information needs to be provided in order to describe the model entirely.

In particular, the set of component references $\mathbf{D}$ must to be defined. As time and time advancement is crucial not only for the agent model AM, but also for the environment model $ENVM$, the set of component references $\mathbf{D}$ consists of all agent references $\mathbf{A}$ as well as environmental services, currently primary data services $\mathbf{PDS}$ and secondary data services $\mathbf{SDS}$.

$$\mathbf{D} = \mathbf{PDS} \cup \mathbf{SDS} \cup \mathbf{A} \tag{4.8}$$

In order to ensure compatibility, each component $d \in \mathbf{D}$, that is each service or agent entity presented in Section 4.3, must at least consist of all structural elements of a component as formally described in Equation 3.1. Otherwise, it would not be possible to integrate them in a discrete-event timely synchronized system.

With Equation 3.3-3.5 and the preceding additional specifications, the developed DEVN and hence the ABM time advancement concept in SIMONA is fully described in a general formal way. Having this general, static description of time in SIMONA, the next question that needs to be answered is how the actual dynamic interaction of multiple components or agents and services respectively can be synchronized in order to reach a stable state during overall model execution. This includes answering the question about the actual time advancement mechanism of the formal concept.

As already mentioned in Section 3.2.3, a corresponding execution mechanism is required, which takes care of component execution as well as time synchronization. Within the next section, this execution mechanism, in the form of a coordinating entity call *SimS*, is presented. Its introduction is followed by a description of the scheduling mechanism and the time synchronization interaction protocol.

4.4.2 Scheduler Service Entity

Entity Structure

In SIMONA, each agent and service entity independently exists within the model. However, to provide a closed, meaningful model environment, logical model time needs to be synchronized to ensure that each entity, may it be an agent or a service, exists in the same environment at the same point of logical time. This synchronization of all components and the handling of external events arriving at the network inputs is the task of the *SimS*. In the current version of the model, there is always only one instance of the *SimS* available, which makes it a coordinator and a root coordinator at the same time. Consequently, the *SimS* is responsible not only for the coordination of all components but also for the overall model execution loop.

Since SIMONA is still an agent-based model, it should be emphasized at this point that the *SimS* is only a coordinating, but not an intervening instance. In concrete terms, this means that the agents can continue to make their decisions autonomously and independently, including the decision about the exact time an activity should take place, which in DEVS terms results in an internal event. The *SimS* merely ensures that the activities of all agents that should take place at a point in logical time t are executed precisely at this point in time. This statement is true for internal and external events from a components perspective. Additionally, the *SimS* is responsible for maintaining the logical stability of the system and ensuring that no agent attempts to perform an activity in the past.

From an ABM perspective, the described time advancement mechanism is denoted with $\mathbf{T}$. Within this mechanism, the concrete logical time representation is modeled by "ticks", and $t \in \mathbf{T}$ represents a single tick. The smallest possible difference Δt between two ticks is exactly $1\,\mathrm{s}$ in physical time. Thus, in the developed model, $1\,\mathrm{s}$ also represents the smallest possible time unit at which an event can occur. By convention, tick 0 is always the first tick and hence the beginning of logical time. As long as the physical time at $t = 0$ is known, this convention allows the calculation of the corresponding physical time for all t.

Formally, the *SimS* can be described as DEVS itself with the following structure:

$$SimS = \langle \mathbf{X}, \mathbf{Y}, \mathbf{S}, \mathbf{L}, \xi, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta, \varepsilon \rangle \tag{4.9}$$

where

$\mathbf{X}$	is the set of input events the scheduler can process,
$\mathbf{Y}$	is the set of output events the scheduler can produce,
$\mathbf{S}$	is the set of states the scheduler can reach,
$\mathbf{L}$	is a list or queue of events sorted by the events time of occurence,
$\xi : \mathbf{T} \to TM$	is a function that maps the events time of occurence to a bag of corresponding output events
$\delta_{\text{int}} : \mathbf{S} \to \mathbf{S}$	is the internal transition function,
$\delta_{\text{ext}} : \mathbf{Q} \times \mathbf{X} \to \mathbf{S}$	is the external transition function,
$\lambda : \mathbf{S} \to \mathbf{Y}$	is the output function,
$ta : \mathbf{S} \to \mathbb{R}^{+}_{0,\infty}$	is the time advance function,
$\varepsilon_S \in \mathbb{N}$	is the number of ahead ticks allowed for optimistic event scheduling.

All input events that can be processed by the *SimS* are presented in Table 4.3.

Table 4.3: Supported *SimScheduler* input event types

Acronym	Name	Additional attributes
IM	*InitMessage*	–
SSM	*StartScheduleMessage*	if & when schedule should be paused again
STM	*ScheduleTriggerMessage*	to-be-scheduled trigger & component reference
CM	*CompletionMessage*	completed trigger id & an optional bag of new-to-be-scheduled triggers
PFM	*PowerFlowFailedMessage*	–
TER	*Terminated* event	sending component reference

Thus, the set of processable input events X is defined as

$$\mathbf{X} = \{IM, SSM, STM, CM, PFM, TER\}. \tag{4.10}$$

Event types that can be generated and emitted by the *SimS* are shown in Table 4.4, leading to the definition of the output set **Y** depicted in Equation 4.11. The special situation of the *SimS*, being an environmental entity on the one hand and controlling all environment and agent model components on the other hand, makes it necessary to distinguish between two different output event type sets. The first set **RE** consists of so-called *runtime events* indicating the current state of the overall model execution. They are an observable description of the overall model during its execution. These kinds of events are emitted to the simulation infrastructure. The second set **CE** consists of the actual *component events*. This set is required for the synchronization of all components during overall model execution.

$$\begin{aligned}\mathbf{Y} &= \mathbf{RE} \cup \mathbf{CE} \\ &= \{SimSM, SimFM, I, R, IC, SI, D, E\} \cup \{ITM, \{TM_d\}\}\end{aligned} \tag{4.11}$$

$$\begin{gathered}TM_d = (T_d, i_{T_d}, d), \\ TM_d \subseteq X_d \wedge T_d \subseteq X_d, i \in \mathbb{N}^*\end{gathered} \tag{4.12}$$

TM_d as part of the **CE** is a set containing *TriggerMessage*s for specific components $d \in \mathbf{D}$. These messages are used to synchronize the logical time notion of all components in the DEVN. It can be compared with a wrapper or wildcard event type containing a component-specific *Trigger* T_d and its trigger index i_{T_d}. T_d represents an internal event of a component d used for its internal state transitions. All *Trigger* must always include at least the time of their occurrence. Hence, the minimum structure of a *Trigger* always is a tuple such that $T_d = (*, t_{T_d})$.

While *Trigger* are generally component-specific and defined accordingly in the respective components, there are also two generic triggers available, the *InitializeTrigger* IT and the *ActivityStartTrigger* AST, which can be processed by all components. The first one triggers a component's state transition from its initial state *Uninitialized* to its default state *Idle*. The latter is a generic one, which can trigger any internal state change in a component that does not require specific information. Together with the TM, the *Trigger* is the coupling point between all components and the *SimS* enabling a stable time synchronization mechanism.

Following the theoretical approach of DEVS, one part of the core functionality of the *SimS* is defined by a finite set of all possible states **S** it can be in. In conjunction with the respective internal and external transition functions δ_{int} and δ_{ext} as well as the output function λ, it fulfills the requirements of a Mealy-FSM [138] and can be represented accordingly.

Table 4.4: Supported *SimScheduler* output event types

Acronym	**Name**	**Additional attributes**
$SimSM$	*SimulationSuccessfulMessage*	–
$SimFM$	*SimulationFailureMessage*	–
I	*Initializing*	–
R	*Ready*	t
IC	*InitComplete*	init process duration
SI	*Simulating*	t_{start} & t_{end}
D	*Done*	t, duration
E	*Error*	error information
ITM	*IllegalTriggerMessage*	reason of illegality
TM	*TriggerMessage*	T_d, i_{T_d}, d

The set of possible states **S** is defined as follows:

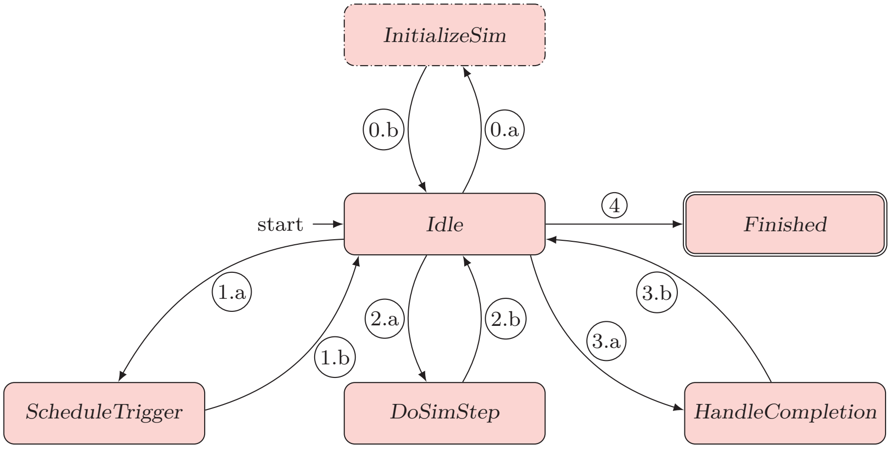
$$\mathbf{S} = \{IDL, INS, DSS, ST, HCM, F\} \tag{4.13}$$

where

IDL	*Idle*, default state,
DSS	*DoSimStep*, schedule execution state,
INS	*InitializeSim*, same as DSS, but restricted to initialization trigger,
ST	*ScheduleTrigger*, component event scheduling state,
HC	*HandleCompletion*, component event execution completion handling,
F	*Finished*, terminate or final state.

The resulting state diagram, given in Figure 4.7a, implicitly includes the transition functions δ_{int} and δ_{ext}, the time advance function ta and the output function λ.

The individual input and output events have been replaced by a number/letter combination to improve readability. The numeric entries do not represent the order of the state transitions and serve only the readability. The INS state can only be reached once during the *SimS*'s lifetime, at the beginning of the overall model execution, and is correspondingly marked with dashed-dotted borders. Within this state, all initialization triggers available to the *SimS* are sent to their receiving components. The corresponding input and output events for each state are shown accordingly in Figure 4.7b. All events are considered to be received and sent in the



(a) *SimScheduler* FSM

#	**Input** $x \in \mathbf{X}$	**Output** $y \in \mathbf{Y}$
0.a	IM	$I \wedge \{TM\}$
0.b	–	IC
1.a	STM	–
1.b	–	ITM on error, – otherwise
2.a	SSM	$\{TM\} \wedge SI$
2.b	–	R
3.a	CM	–
3.b	–	–
4	$- \wedge PFM \wedge T$	$(D \vee E) \wedge (SimSM \vee SimFM)$

(b) Input and output events of the different *SimScheduler* states

Figure 4.7: FSM model of the *SimScheduler* service

form of an event message by the scheduler. Each state can only be reached if a corresponding message has been received from the *SimS*.

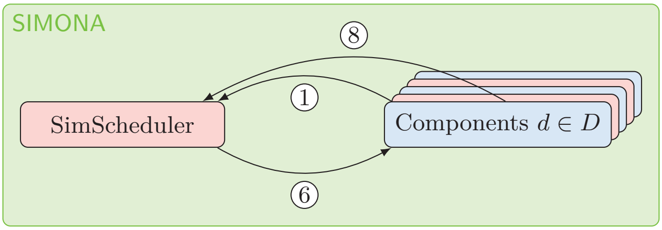
The only exception is the *Idle* state, to which it returns after each successful processing of a received event. This is indicated with a "–". For example, if the *SimS* is in the *Idle* state and receives an IM message a state change into *InitializeSim* takes place. In doing so, the *SimS* emits an I event as well as a set of TM events as part of the *InitializeSim* state. After the completion of all process steps in *InitializeSim*, the *SimS* switches back to *Idle* and emits an IC event during the transition.

In addition to the already familiar form of DEVS, the *SimS* incorporates a list $\mathbf{L} = \{TM_d\}$ and a function ξ which are both required in order to synchronize the components. L is a list or a queue, also called *TriggerQueue* containing component events TM to be scheduled. ξ is a mapping function, also called *AwaitingResponses* that maps a bag of scheduled TM to their occurrence time. Both elements will be discussed more in detail in the following as part of the scheduling mechanism.

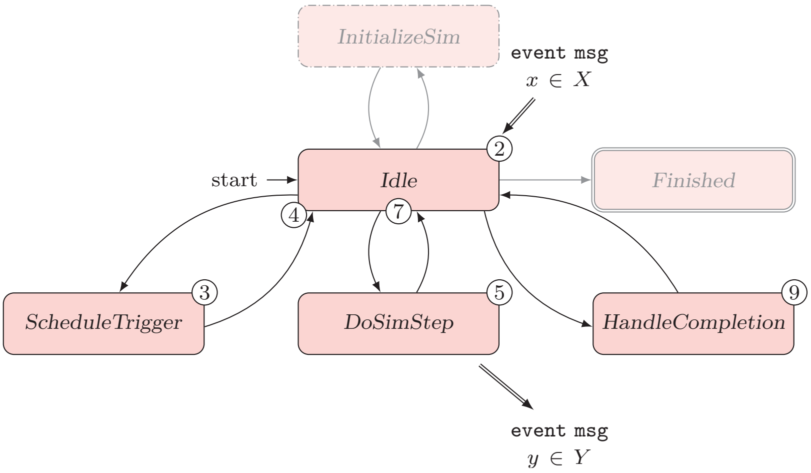
Scheduling Mechanism and Entity Interaction

Within the modeled system, the global logical time $t_\mathrm{g} \in \mathbf{T}$ is defined as the point in time the *SimS* is currently in. Whenever a component wants an activation at a certain point in time, it sends an STM containing the desired activation tick as well as the trigger that should be scheduled to the *SimS*, which in turn schedules the trigger to send it out at the requested tick to the requesting entity. However, when using an optimistic event scheduling strategy, it might be possible that the *SimS* is ahead of time compared to the components, which therefore may receive and process messages in the wrong order and the overall system gets out of sync. As a consequence, these components would return STM with $t_{T_d} < t_g$, which would lead to an invalid state of the system. This issue is tackled in SIMONA by limiting the acceptable time horizon for optimistic scheduling using the already introduced time ahead parameter ε. It can be used to enable or disable a conservative or optimistic scheduling strategy. As a result, the system is considered to be in a time-synchronous state whenever the $t_g - \varepsilon \leq t_{T_d} \leq t_g$ of a received STM is fulfilled.

With this definition of global time, the underlying time synchronization algorithm, the event scheduling algorithm, is now describable. In order to perform its central task, the coordination and logical time synchronization of all components $\mathbf{D}$, the *SimS* requires specific information about the time of occurrence of external and internal events of each component $d \in \mathbf{D}$. Figure 4.8 gives an overview of the scheduling mechanism and the interaction protocol between the *SimS* and the components.



(a) *SimScheduler* interaction protocol



(b) *SimScheduler* algorithm state transitions

Figure 4.8: General *SimScheduler* entity interaction protocol

The main steps of the scheduling mechanism, executed by the *SimS* in order to advance the global overall model time t_g mainly consists of eight steps.

① Each time a component intends to perform an internal state transition at a time t_{T_d}, it has to announce this state transition by sending a $STM(t_{T_d}, T_d)$ to the *SimS*.

② Internally, the *SimS* receives the STM in its *Idle* state.

③ The received STM causes a state transition into the *ScheduleTrigger* state where the message is processed. Every time, the *SimS* receives an STM a validitiy check according to Equation 4.14 is carried out.

$$v(STM) = \begin{cases} ITM, & \text{if } t_{T_d} < t_{\text{g}} - \varepsilon \\ TM, & otherwise \end{cases} \tag{4.14}$$

If the validation fails, an ITM is send back to the component, informing it about the invalid *Trigger* time. If the output is valid, the resulting TM is added to the *TriggerQueue* L. Before switching back to *Idle*, L is sorted by the *Trigger* event times t_{T_d} in ascending order and, if t_{T_d} is equal for two or more components, by their trigger id i_{T_d}.

④ Back in *Idle* the *SimS* either waits for a SSM or proceeds with the schedule execution by switching to *DoSimStep*. If the schedule execution has already started, it switches to *DoSimStep* without waiting for another message.

⑤ In *DoSimStep* the *SimS* continues the schedule execution. This process starts with a check if, it is possible to send out new TM without violating system synchronization conditions according to Equation 4.15.

$$advance = \begin{cases} \text{true}, & \text{if } \xi(t)|_{\forall t < t_{\text{g}}} = \emptyset \vee t_{\text{g}} - t_{\text{oldest}} \leq \varepsilon \\ \text{false}, & \text{otherwise} \end{cases} \tag{4.15}$$

Informally, the *SimS* is allowed to send out more TM if either the *AwaitingResponses* mapping function ξ is empty or if the oldest, that is the TM with the smallest tick t_{T_d} such that $t_{T_d} = t_{\text{oldest}} = \min\{t|\xi(t) \neq \emptyset\}$, subtracted from the global time t_{g} is smaller or equal to the allow ahead time ε. If *advance* is false, the *SimS* switches back to *Idle* and waits for more CM which would allow further time advancement. If *advance* is true, the global time t_{g} is set to the time t_{T_d} of the first TM in L and sends out all TM in L where $t_{T_d} = t_{\text{g}}$. Each TM_d that is send out is also stored internally using the mapping function ξ which maps the *Trigger* time t_{T_d} to a bag of output events TM_d with the same *Trigger* time. This procedure is executed in a loop until Equation 4.15 cannot be satisfied anymore.

⑥ As a result of the *DoSimStep* procedure, components receive their TM including the previously requested *Trigger* for internal state transitions.

⑦ If Equation 4.15 cannot be satisfied anymore, the *SimS* switches back from *DoSimStep* to *Idle*. Under normal conditions, it could be possible that new STM messages arrive, which would trigger a state transition to *ScheduleTrigger* as in ③. However, in this example it is assumed, that now new STM are issued by the components.

⑧ After finishing their internal state transition, each component $d \in D$ that has received a TM_d now replies with a CM, indicating that all transitions they wanted to execute at the current tick t_g have been carried out.

⑨ In the final step of the interaction protocol, the *SimS* extracts the *Trigger* id i_{T_d} from each received CM as well as the time information the *Trigger* has been issued. Using this information, the corresponding TM is removed from the *AwaitingResponses* map ξ.

All steps from ① to ⑨ are executed repeatedly until the global time t_g reaches the defined termination time $t_\mathrm{ter} = \max(\mathbf{T}) + \varepsilon$.

4.4.3 Summary

This section presents SIMONA's scheduling mechanism and the executing entity, the *SimS* including the interaction protocol between the *SimS* and the individual simulation components, services, and agents. In addition to the presented mechanism, the following remarks should be taken into account:

– The presented protocol is valid for the general scheduling and the overall model initialization. The only difference is that during the INS state, only *Trigger* specifically used for component initialization are send from the scheduler and the schedule execution is paused afterward.

– Even if the scheduling of *TriggerMessages* TM in ⑤ and ⑥ is executed sequentially, depending on the implementation, the *Trigger* processing as internal events inside the components may be concurrent and asynchronous.

– The scheduler does not restrict the general interaction between agents and services concerning the agent and service interaction. The interaction between all components with the help of related external events is still possible. The only requirement is that the time t of sending and receiving the respective event must be within the global time interval $tg - \varepsilon$.

4.5 Agent-based Distributed Backward-Forward Sweep Algorithm

The objective of the developed simulation model is the generation of realistic time series of the electric power system at the distribution level. In addition to power feed-in and consumption of the different assets connected to the grid, this includes the time series of the electric grid utilization and the grid state. In the following, Subsection 4.5.1 describes the specifications that the agents representing the grid need to fulfill. Afterward, Subsection 4.5.2 deals with the mathematical description of the theoretical idea behind the applied grid partition scheme. Finally, Subsection 4.5.3 and 4.5.4 detail the agent representing the whole grid and the communication and negotiation protocol between multiple *GAs*.

4.5.1 Power Flow Algorithm and Grid Model Specifications

A crucial part of SIMONA is the electricity grid model, representing the power distribution system. To calculate the grid power flow, the agent-based nature of the simulation model and the objective to be as flexible as possible regarding the supported grid structure leads to several requirements, summarized in Table 4.5.

Table 4.5: Power flow algorithm and grid model specifications overview

Power Flow Specifications
Support for single grids
Support for multi grids with adjacent grids as point equivalent
Support for multi-voltage grids with detailed grid topologies
Applicability in the context of an agent system
Scalability up to several thousand nodes
Grid Model Partition Scheme Specifications
Applicability on single, multi and coupled grid topologies
Support for subgrid distribution on multiple computational nodes

The first set of requirements relate to the capabilities of the power flow calculation concerning the grid structure. The applied approach must calculate the power flow for single, galvanically isolated low-voltage grids where each load or feed-in asset is modeled explicitly. Moreover, it must also be able to handle cases where a single point load or feed-in represents adjacent downstream low-voltage grids. The capability to simulate coupled, multi-voltage level grids from the low-voltage level up to the high-voltage level completes the power flow specifications. In this case, each single, galvanically isolated grid may be modeled with all connected assets explicitly or partly as single point load or feed-in. Concerning the agent-based model, it is crucial that the selected power flow calculation is applicable in a distributed context. The system's decentralized structure implicitly requires that the power flow approach can handle distributed data and knowledge. Finally, the approach needs to be scalable up to several thousand grid nodes. With an increasing grid size, particularly in the case of coupled grids in a multi-voltage level simulation, the computational complexity increases. Therefore, concerning the performance and scalability of the selected approach, a suitable and efficient procedure must be chosen for the grid partioning and power flow execution for multi-voltage level grids. At the same time, the grid model should be sufficiently detailed, both for individual, galvanically isolated grids and for grids spanning several voltage levels.

Taking these requirements into account, one way to determine the power flow is to apply a backward-forward sweep algorithm as done in [139]. In its general structure, this algorithm suits quite well for an agent-based approach as it allows the decomposition of the overall power flow equations into several smaller partial equations. These equations can then be assigned to several agents with limited, decentralized knowledge. Each agent then solves its partial equation individually, and the individual results are subsequently aggregated to the overall solution. Depending on the actual algorithm implementation, the partioning of the overall grid into several subgrids can be chosen flexibly.

Based on the requirements, the boundary for a single grid partition is defined at its galvanic isolation point, that is, at the grid coupling point to the respective adjacent higher- or lower-level grid. This partioning approach provides a good tradeoff between single grid and coupled grid calculation performance as specifications require. In particular, if the agents are implemented to allow concurrent execution, this approach enables the power flow computation of all subgrids at the same voltage level in parallel. Furthermore, this approach allows us to tackle the single grid and the coupled grid case with only one model for the *GA*.

From the performance and scalability perspective, the modeled and implemented overall grid model and the *GA* interaction is a distributable system. Distributable in this context means that the algorithm does not rely on a closed, centralized solution but is executed in a distributed fashion by all *GAs* involved. This distributed approach enables the algorithm's execution on multiple, different computational nodes, which correspondingly allows efficient utilization of computational clusters.

This section describes the partitioning process, the agent models and the solution algorithm in detail, based on the formulated specifications. For all subsequent considerations, a single-phase, symmetric and balanced grid structure is assumed.

4.5.2 Mathematical Model of Single and Coupled Subgrids

Single subgrid

A single subgrid in this work refers to an electricity grid that exists as a galvanically isolated grid on a single voltage level, meaning that all grid elements are operated with the same voltage level except for the high-voltage nodes of transformers. Examples of single subgrids are individual, galvanically separated low-voltage grids or medium-voltage grids, representing subordinate grids only as point load or feed-in and not in their detailed form. Furthermore, it is assumed that a single subgrid always contains at least one transformer that is connected on the high-voltage side to a static external grid. The high-voltage node of the transformer then serves as a balancing slack node for the subgrid. The mathematical representation of individual subgrids thus essentially corresponds to the equations presented in Section 2.

As a reminder, the overall grid g can be described with the set of all nodes $\mathbf{N}$ and the node admittance matrix $\left[\underline{Y}_{\mathrm{N},g}\right]$. The set of all nodes is equal to the sum of all *pq*, *pv* and *slack node* sets. The node admittance matrix represents the mathematical link between the node current and noded voltage vectors of the grid (Equation 4.16).

$$\underline{\vec{i}} = \left[\underline{Y}_{\mathrm{N},g}\right] \cdot \underline{\vec{v}} \tag{4.16}$$

Coupled subgrids

Coupled subgrids in this work consist of two or more galvanically separated subgrids of different voltage levels interconnected by one or more transformers. Thus, an overall grid consisting of coupled subgrids is, for example, a medium-voltage grid with one or more low-voltage subgrids, which are not represented as point models, but whose topology is modeled explicitly as a single subgrid. Furthermore, each subgrid in a coupled grid contains at least one transformer connected to a static grid or the coupled upstream grid at the higher voltage side. The high-voltage node of the transformer, either located in the static or upstream grid, then serves as the balancing slack node for the corresponding subgrid.

The entire grid, coupled across voltage levels, thus consists of a set of grid partitions or subgrids $\mathbf{G} = 1, 2, 3, ..., g$. Since g represents a single subgrid, it can be described using the already presented subgrid equations in a slightly modified form.

The set of all nodes in a single subgrid $g \in \mathbf{G}$ correspondingly is defined as

$$\mathbf{N}_g = \mathbf{N}_{\text{pq},g} + \mathbf{N}_{\text{pv},g} + \mathbf{N}_{\text{s},g} \tag{4.17}$$

Consequently, the subgrids characterizing the relationship between node currents, node voltages and grid element admittances becomes

$$\tilde{\vec{\underline{i}}}_g = \left[\underline{Y}_{\text{N},g}\right] \cdot \vec{\underline{v}}_g \tag{4.18}$$

However, in contrast to a single subgrid, the node current vector $\tilde{\vec{\underline{i}}}_g$ of coupled subgrids is composed of two separate parts.

$$\tilde{\vec{\underline{i}}}_g = \vec{\underline{i}}_g + \vec{\underline{i}}_{\text{con},g} \tag{4.19}$$

The first vector $\vec{\underline{i}}_g$ represents the current from all system participants connected to the corresponding nodes within the subgrid g. The second one, $\vec{\underline{i}}_{\text{con},g}$, is the vector of all equivalent current injections from all adjacent downstream grids, correctly assigned to their respective node n in the set of coupling nodes $\mathbf{D}_g$. Consequently, each element $\underline{i}_{\text{con},g,n}$ is defined as the sum of all current injections resulting resulting from adjacent downstream grids connected at a node n.

$$\vec{\underline{i}}_{\text{con},g} = -\sum_{\forall d \in \mathbf{D}_g} \vec{\underline{i}}_{\text{con},d} \tag{4.20}$$

The negative sign in Equation 4.20 is required because the current injection of a downstream subgrid has an opposite effect from the adjacent upstream grid's perspective. In other words, if the slack node of a downstream grid provides a positive current injection to the downstream grid, from the perspective of the adjacent upstream grid, where the slack node is located, this current injection is negative.

In order to correctly assign the downstream slack nodes to their respective upstream connection nodes, the set of all nodes within g that are coupling points to the adjacent downstream grid d is defined as

$$\mathbf{D}_g = \mathbf{N}_g \setminus \mathbf{N}_{\mathrm{s},g} \cap \mathbf{N}_{\mathrm{s},d} \,|\, d, g \in \mathbf{G} \wedge d \neq g \tag{4.21}$$

Equation 4.20 can be represented in a simplified way, taking into account the node mapping, using a mapping matrix $\delta_{g,d}$ that maps the current injection at slack nodes for each downstream grid d to their connecting counterparts in the upstream grid g as follows:

$$\underline{i}_{\mathrm{con},g,n} = - \sum_{d \in \mathbf{D}_g} [\delta_{g,d}] \cdot \underline{i}_d \tag{4.22}$$

with

$$[\delta_{g,d}] = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 1 & \dots \\ 0 & 0 & 0 & \\ & \vdots & & \ddots \end{bmatrix}. \tag{4.23}$$

It is important mention the implications of possible adjacent upstream subgrids of g to complete the entire picture effectively. In addition to the current injection of all adjacent downstream grids into a specific subgrid g, the influence of all upstream grids of g on the transformer's high-voltage nodes, the slack node, needs to be taken into account. If g has no further upstream grids and all high-voltage transformer nodes are located in a static grid, this grid provides the voltage magnitude and angle. However, in case that one or several adjacent upstream grids of g are subgrids by themselves, the slack voltages for all high-voltage transformer nodes need to be provided by the corresponding adjacent upstream grids. This does not change the describing grid equations, as the approach for a static upstream grid and a detailed single subgrid are equal. Though, it does impact the agents' communication and interaction protocol and is relevant for the following agent entity part.

The preceding description of a single subgrid in an overall grid of coupled subgrids can now be used to compose a voltage level spanning grid based on galvanically isolated subgrids. The resulting grid description is given in Equation 4.24, where $[\mathrm{Id}_g]$ is the identity matrix, $\left[\underline{Y}_{\mathrm{N},g}\right]$ is the node admittance matrix, $\underline{\vec{v}}_g$ is the node voltage vector and $\underline{\vec{i}}_g$ is the node current vector of subgrid g.

$$[\mathrm{Id}_g] \cdot \begin{pmatrix} [\underline{Y}_{\mathrm{N},0}] \\ [\underline{Y}_{\mathrm{N},1}] \\ \vdots \\ [\underline{Y}_{\mathrm{N},g}] \end{pmatrix} \cdot \begin{pmatrix} \vec{\underline{v}}_0 \\ \vec{\underline{v}}_1 \\ \vdots \\ \vec{\underline{v}}_g \end{pmatrix} = \begin{pmatrix} \vec{\underline{i}}_0 - \sum_{d \in D_0} \left([\delta_{0,d}] \cdot \vec{\underline{i}}_d\right) \\ \vec{\underline{i}}_1 - \sum_{d \in D_1} \left([\delta_{1,d}] \cdot \vec{\underline{i}}_d\right) \\ \vdots \\ \vec{\underline{i}}_g - \sum_{d \in D_g} \left([\delta_{g,d}] \cdot \vec{\underline{i}}_d\right) \end{pmatrix} \tag{4.24}$$

The structure of the composed overall grid, that is, the coupling connections between the galvanically separated subgrids can be depicted illustratively by contracting the elements of each subgrid to a single point. As a result, a topology graph emerges, which describes the neighborhood relations between the subgrids. Depending on the meshing degree and the overall grid's topology, the result is a tree or tree-like structure.[10] Each vertex of the graph represents a single subgrid, and the connecting edges represent the respective voltage transformation level. An exemplary subgrid topology graph for a radial overall grid is given in Figure 4.9.

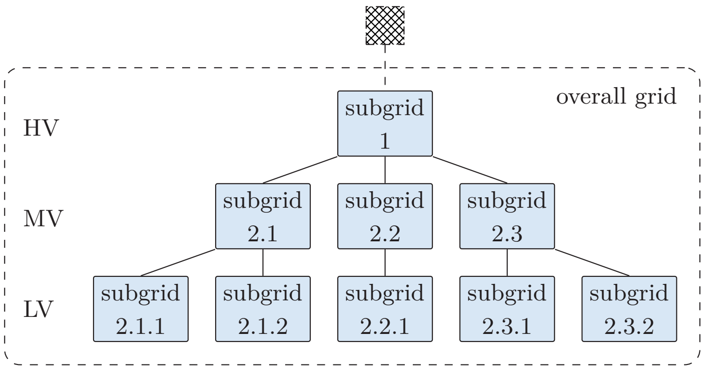


Figure 4.9: Exemplary multi-voltage level subgrid topology graph

In order to determine the overall grid state for different operation points, it is not necessary to calculate the equations for the whole grid. Instead, it is sufficient to solve the equations for each subgrid, taking into account node current injections from adjacent downstream and voltage information from adjacent upstream grids, as the overall grid is composed of single subgrids with coupling information. If a large, extensive grid is given, it can be decomposed into several smaller subgrids. As a result, it is possible to efficiently split up an extensive overall grid into several

[10] In this context, a tree refers to the mathematical description in graph theory.

subgrids and assign a subgrid to each agent responsible for the subgrid calculations. As evident from the exemplary subgrid topology graph, it is also possible to execute subgrid power flow calculations which are independent of each other in a concurrent, highly efficient way. For example, in Figure 4.9 it is possible to calculate the grid state of all grids at the same voltage level in parallel, as they do not depend on each other, but only on their adjacent downstream and upstream grids.

4.5.3 Grid Agent

Together with the *SPA*, the *GA* represents central agent entities of SIMONA. It consists of a physical model and a behavioral model and hence can be classified as Type 2 Agent. The main task of the grid agent in the current modeling is to solve the power flow equations and determine the current state of the grid assigned to it. To do so, it must interact with all *SPAs*, connected to the grid for which it is responsible. The actual number of agent entities available in the model depends on the investigated grid topology. In general, one *GA* is responsible for a single subgrid. If the overall grid consists of a system of coupled subgrids, the individual grid agents must interact not only with the *SPAs* in their subgrid but also with the *GAs* representing the adjacent upstream and downstream subgrids.

Physical Model

As depicted in Figure 4.10, the physical model of a *GA* consists of the corresponding subgrid topology it is responsible for, as well as the algorithm used to determine the grid state. While the underlying topology model is exchangeable, it is currently represented by a graph consisting of electrical nodes as vertices and lines, switches and transformers as connecting edges.

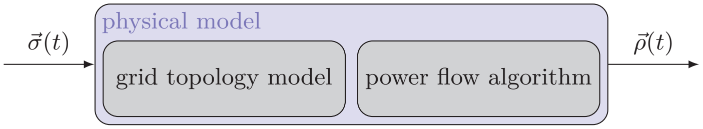


Figure 4.10: Physical model of the *GridAgent*

In addition to the topological information, the *GA* requires information about the nominal apparent power $S_{\mathrm{n},g}$ in p.u. as well as the nominal voltage $V_{\mathrm{n},g}$ in p.u. of all nodes $n \in \mathbf{N}_g$. This information is provided from all *SPAs* during DBFS execution. For correct execution of the DBFS, the following configuration parameters are required for the physical model of the *GA*:

- convergence threshold $\varepsilon_{\text{sweep}}$ for the outer loop;
- convergence criterion ε_{nra} for the internal Newton-Raphson Method (NR-Method) power flow calculation;
- maximum sweep counter $i_{\text{sweep,max}}$.

A power flow calculation needs to be executed on the physical model to determine the grid state. While in this thesis, a NR-Method is used for this, the specific power flow algorithm is easily interchangeable.

Behavioral model

The behavioral model of the agent consists of two inseparable parts. The first part is the state of the agent. The second, *GA* specific part holds the decision function $f(\cdot)$ and allows the agent to make decisions and take actions based on its state. Like the physical model, the behavioral model also requires several parameters. First of all, the power flow interval t_{pf}, used to execute a power flow calculation every t_{pf} ticks, needs to be provided. Furthermore, in order to communicate with all *SPAs* connected to its grid, the agent requires additional information about all *SPA* references connected to each node of its subgrid. If the overall grid model is a system of coupled subgrids, in order to communicate with its adjacent downstream and upstream grids, the *GA* also requires the corresponding *GA* references. Both reference sets can be derived from the physical model.

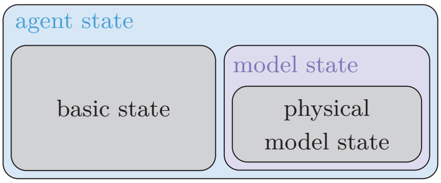


Figure 4.11: *GridAgent* state description

The behavioral model of the agent is closely linked to the respective requirements of a component as explained in Section 3.2, but is slightly more complex than a regular one. The DEVN component part of a *GA* can be described formally as

$$GridAgent = \langle \mathbf{X}, \mathbf{A}, \mathbf{S}, \mathbf{M}, \mathbf{Y}, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle \tag{4.25}$$

where

$\mathbf{X}$	is the set of input events supported by the *GA*,
$\mathbf{A}$	is the set of agent states,
$\mathbf{S}$	is the set of basic states,
$\mathbf{M}$	is the set of model states,
$\mathbf{Y}$	is the of output events supported by the *GA*,
$\delta_{\text{int}} : \mathbf{A} \rightarrow \mathbf{A}$	is the internal transition function,
$\delta_{\text{ext}} : \mathbf{A} \times \mathbf{X} \rightarrow \mathbf{A}$	is the external transition function,
$\lambda : \mathbf{A} \rightarrow \mathbf{Y}$	is the output function,
$ta : \mathbf{A} \rightarrow \mathbb{R}^{+}_{0,\infty}$	is the time advance function.

The *GA* supports a variety of different input and output events, which are required depending on the particular agent state and the underlying physical grid model. For reasons of clarity, a comprehensive introduction of all supported events is omitted here. In the course of this chapter, individual events will be introduced if necessary for further understanding. Additionally, Appendix A.1 provides a comprehensive overview of all supported events, including their acronym. To avoid confusion, *GA* event acronyms are not listed in the textual abbreviations, but exclusively in Table A.1 - A.3.

The agent state is defined as a tuple of the basic state and the actual state model state, while the basic state determines the actual agent state name. As a result, the determining agent state set A, required for all transitions, is defined as

$$\mathbf{A} = \{(s, m) | s \in S, m \in M\} \, . \tag{4.26}$$

The basic state set **S** of a *GA* is defined as

$$\mathbf{S} = \{UI, IDL, SG, HPFC, CPD, F\} \tag{4.27}$$

where

UI	*Uninitialized*,
IDL	*Idle*, default state,
SG	*SimulateGrid*,
$HPFC$	*HandlePowerFlowCalculations*,
CPD	*CheckPowerDifferences*,
F	*Finished*, terminate or final state.

The model state set **M** of a *GA* is defined as follows:

$$\mathbf{M} = \{B, PFD\} \tag{4.28}$$

where

B	*BaseModelState*,
PFD	*PowerFlowDoneState*.

The resulting FSM with neglected input and output events of the *GA* which implicitly includes the agent's transition functions is given in Figure 4.12.

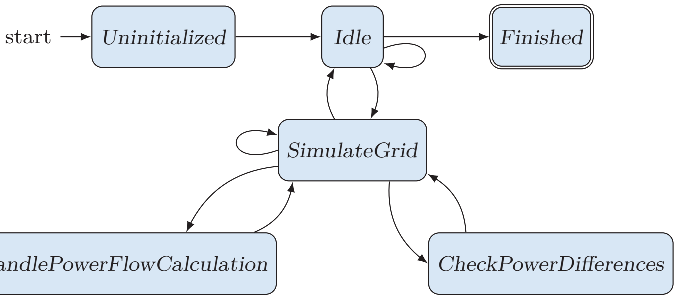


Figure 4.12: *GridAgent* behavioral model FSM

The FSM of the *GA* consists essentially of 3 generic states as well as 3 agent-specific states. The generic states include *Uninitialized*, *Idle* as well as *Finished*. The state

Uninitialized is the state immediately after the start of the agent. In this state, the agent expects its initialization data. All other messages it receives in this state are deferred until the initialization is complete. After the initialization, the agent switches to its default state, *Idle*. It stays in this state until the state of its subgrid needs to be determined. The *Finished* state is the state in which the agent performs de-initialization operations, such as a release of resources before it terminates. The three agent-specific states *SimulateGrid*, *HandlePowerFlowCalculations* and *Check-PowerDifferences* are required to perform a DBFS. Here, *SimulateGrid* represents the basic state in the context of DBFS, into which the agent switches from the *Idle* state in order to perform a grid calculation. It is used to communicate with all *SPAs* located in its subgrid as well as with adjacent upstream and downstream *GAs*. When all grid calculation relevant data is available, the agent switches to the *HandlePowerFlowCalculation* state and performs power flow calculation using the received data and its physical model. If any voltage-regulated loads or feeders are located within its subgrid, additional iteration loops with the corresponding *SPAs* take place based on the power flow results. For this purpose, the *GA* switches back and forth between *HandlePowerFlowCalculation* and *SimulateGrid*. The third agent-specific state *CheckPowerDifferences* is only relevant for most superior or root *GA* which represents the static external grid. Within this state, the respective *GA* checks whether there is a convergence of the overall grid or whether additional iteration loops are necessary to determine the overall grid state.

4.5.4 Solution Algorithm and DBFS Agent Interaction Protocol

In order to determine the state for a grid that can be split into several subgrids as explained in Subsection 4.5.2 the algorithm depicted in Figure 4.13 can be used. The algorithm relies on the convergence of local power flow calculations executed by each subgrid within its physical model. If one or multiple local power flows cannot be solved, the overall grid state cannot be determined accordingly. In contrast to existing algorithms in the literature, the presented approach works on a “request” base. While existing algorithms usually start with a backward sweep phase from the leaves of a subgrid topology graph, the developed algorithm starts with an initial request forward sweep phase. This phase is necessary in an agent-based setting where agents only have limited information about their adjacent upstream and downstream grids. It differs from all following phases in the sense that no local power flow calculations are executed. Instead, the agents request the required information about equivalent loads or feed-ins from their adjacent downstream grids to indicate that they rely on this information. As long as the leaves of the subgrid topology graph are not reached, none of the upstream *GAs* is able to provide this information. When the leaves are reached, the corresponding *GAs* execute their first local power flow calculation. The power flow calculation is possible, as leaf *GAs* do not rely on any equivalent loads or feed-ins from adjacent downstream subgrids. Although this first request sweep phase is characteristic for the presented approach, to stay congruent with existing literature, it is also called "backward-forward" sweep algorithm. The naming can be furthermore justified by the fact that the methodology and underlying idea of the algorithm are still similar while being extended to be applicable in an agent-based setting.

The algorithm starts by bringing all involved *GAs* into their *SimulateGrid* state. Additionally, each agent adapts its physical model and sets the voltage magnitude of all nodes to p.u.. Internally, also their sweep iteration counter i_{sweep} is initialized or reset to zero. Afterward, the first request for equivalent loads or feed-ins from adjacent downstream *GA* is sent out, followed by a response of the coupling node voltage to all requesting adjacent downstream grids. This process is repeated until the leaves of the subgrid topology graph are reached. When the leaves are reached, the backward sweep phase starts and all leaf subgrids calculate their local power flow using their physical model. The request for equivalent load or feed-in from the adjacent upstream grid is then answered based on the power flow results. Subsequently, the upstream grid uses the results from the downstream grid and executes its own local power flow to provide the requested equivalent load or feed-in. This process is repeated until the root or external grid *GA* is reached. When reached, the root coordinator executes a convergence check. First of all, it checks if i_{sweep} is bigger than zero. If not, a new sweep is started. If yes, it checks if the power flow results of the overall grid are converging according to Equation 4.29 or if the maximum number of allowed sweeps $i_{\text{sweep,max}}$ is reached. If the overall power flow is not converging or the maximum number of sweeps is not yet reached, another forward-backward sweep is initiated. If the number of maximum sweeps is reached, no valid result can be found. If the overall grid is converging, a feasible solution has been found. In both cases the algorithm terminates.

$$\begin{aligned}
\left|\left|\sum p_{\text{L},i}^{(i_{\text{sweep}}-1)} - \sum p_{\text{G},i}^{(i_{\text{sweep}}-1)}\right| - \left|\sum p_{\text{L},i}^{(i_{\text{sweep}})} - \sum p_{\text{G},i}^{(i_{\text{sweep}})}\right|\right| &< \varepsilon_{\text{sweep}} \\
\left|\left|\sum q_{\text{L},i}^{(i_{\text{sweep}}-1)} - \sum q_{\text{G},i}^{(i_{\text{sweep}}-1)}\right| - \left|\sum q_{\text{L},i}^{(i_{\text{sweep}})} - \sum q_{\text{G},i}^{(i_{\text{sweep}})}\right|\right| &< \varepsilon_{\text{sweep}}
\end{aligned} \tag{4.29}$$

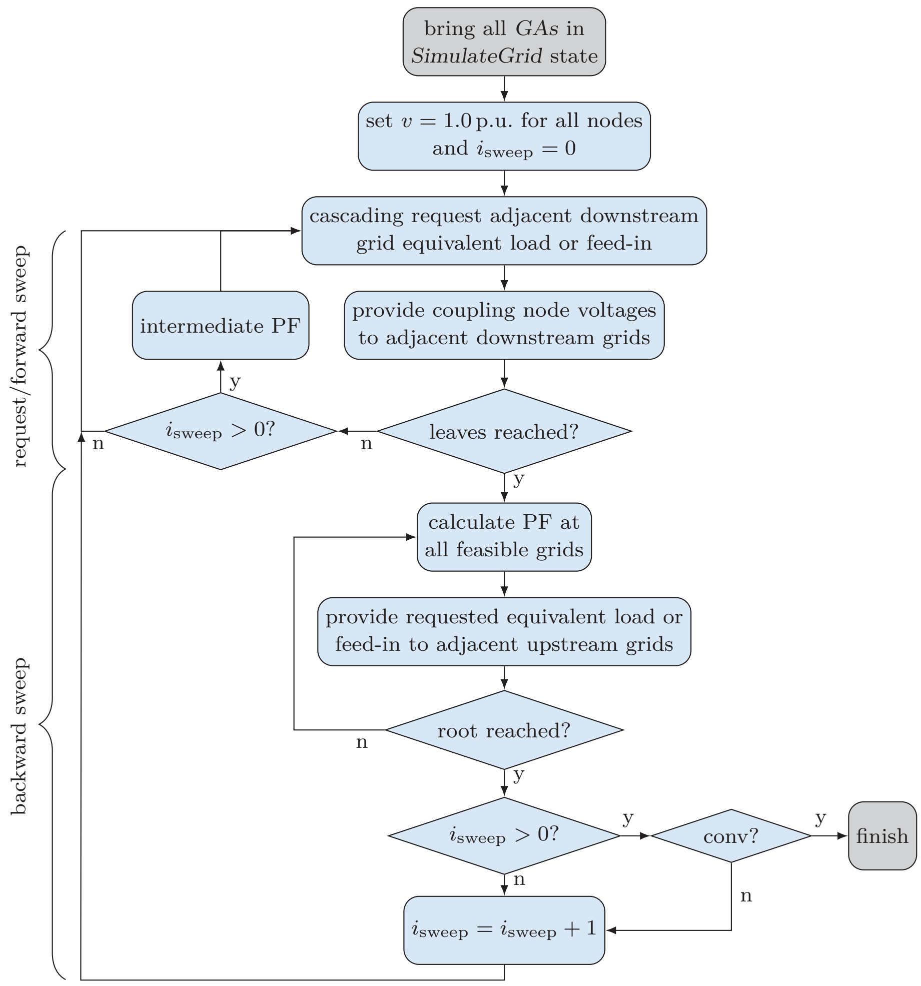

bring all *GAs* in *SimulateGrid* state
set $v = 1.0\,\text{p.u.}$ for all nodes and $i_{\text{sweep}} = 0$
cascading request adjacent downstream grid equivalent load or feed-in
intermediate PF
provide coupling node voltages to adjacent downstream grids
$i_{\text{sweep}} > 0$?
leaves reached?
calculate PF at all feasible grids
provide requested equivalent load or feed-in to adjacent upstream grids
root reached?
$i_{\text{sweep}} > 0$?
conv?
finish
$i_{\text{sweep}} = i_{\text{sweep}} + 1$
y
n
request/forward sweep
backward sweep

Figure 4.13: Agent-based DBFS flow chart

Based on the description of the overall algorithm, the agent communication protocol can be introduced. For this protocol, the global algorithm must be translated into an interaction protocol for the involved *GAs* that produces stable results. In contrast to the general algorithm, where all information is centrally known, this communication protocol needs to consider agents' limited knowledge and distributed nature.

Within the developed protocol, three different behavioral modes or *GA* roles are required. First of all, there is the root or external static grid role, which needs to be fulfilled by the *GA* located at the root of the subgrid topology graph. It is responsible for the overall grid convergence check. Then, there is the role of all *GAs* that are vertices but not leaves in the subgrid topology graph. These agents rely on adjacent downstream grid equivalent load or feed-in data. Finally, there is the role of all *GAs* that are leaves of the subgrid topology graph and hence do not rely on information about equivalent loads or feed-ins from adjacent downstream grids.

The communication protocol is exemplarily depicted in Figure 4.14. Within the protocol, a star ★ indicates the execution of a local power flow calculation. A black square ■ indicates an overall convergence check according to Equation 4.29. The sample overall grid consists of one external static grid (*GA* 0), one medium-voltage grid (*GA* 1) and two low-voltage grids (*GA* 2/3) connected to the medium-voltage grid. The interaction of the low-voltage grid is the same for both of them. Hence, their interaction is depicted cumulated to improve readability. While the overall grid consists of only four *GAs* it contains the full, generic communication protocol that can be applied to any number of grid layers.

The entire protocol consists of the following steps:

① The DBFS grid simulation interaction protocol starts with a *TriggerMessage* (*TM*) containing an *ActivityStartTrigger* (*AST*) that is send out by the *SimS* to all *GAs*.

② The *AST* is processed in the *GAs Idle* state and triggers a state transition into its *SimulateGrid* state. When reaching this state, each *GA* starts requesting the required values in order to perform their local power flow calculation from the respective other agents.

③ First of all, a *RequestAssetPowerMessage* (*RAP*) is sent to all *SPAs* located in the respective grid area. Subsequently, a *RequestGridPowerMessage* (*RGP*) is sent to all adjacent connected downstream *GAs*. Finally, the slack node voltages, which are specified from one or several upstream grids, are requested through a *RequestSlackVoltageMessage* (*RSV*) to the respective adjacent connected upstream *GAs*. This process cascades down until the subgrid topology

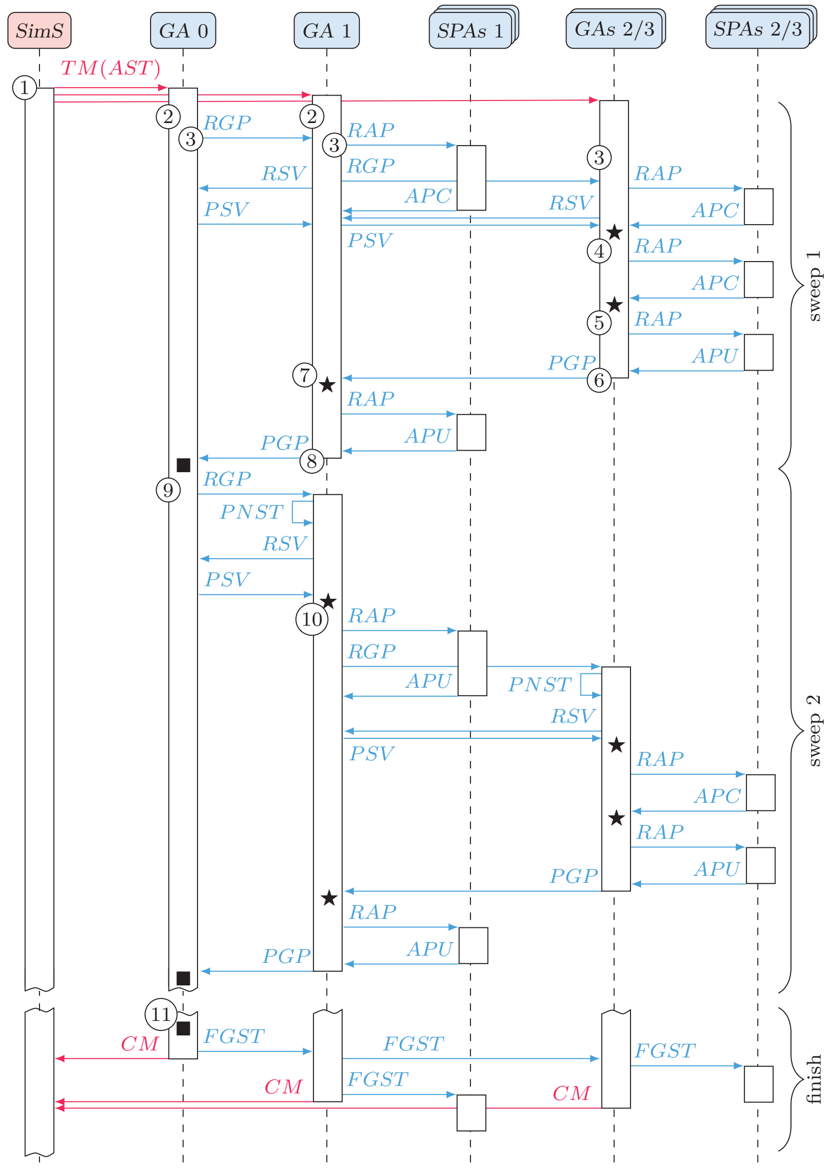


Figure 4.14: DBFS agent interaction protocol

graph leaf *GAs* are reached. Afterward, each *GA* waits until all request messages are answered.

④ As an answer for the *RSV* a *ProvideSlackVoltageMessage* (*PSV*) containing the current slack voltage is send back from all *GAs* with downstream grids to the corresponding adjacent downstream *GAs*. Furthermore, all *SPAs* within each subgrid are responding to the *RAP* with an *AssetPowerChangedMessage* (*ACP*) containing information about load or feed-in of each asset. As the *GAs* located at the leaves of the subgrid topology graph do not rely on equivalent load or feed-in information from adjacent downstream grids, they are the first *GAs* that execute a power flow when they received all answers on their *RSV* and *RAP*. In order to take voltage-regulated assets into account, another *RAP* is sent to all *SPAs* after the first successful power flow with updated voltage information for the node the corresponding asset is connected.

⑤ If at least one asset changes their feed-in, indicated by an *APC* response, another power flow is executed. This iterative negotiation process is repeated until all assets reply with an *AssetPowerUnchangedMessage* (*APU*). In order to avoid non-convergence by grid assets whose regulation is fluctuating up and down, the number of negotiation iterations between the *GA* and its *SPAs* can be limited to a predefined value.

⑥ When a final solution for the local grid state has been found, each *GA* responds to the previously received *RGP* with a *ProvideGridPowerMessage PGP* to the requesting upstream *GA*.

⑦ When all requested information, that is, load or feed-in of all assets and adjacent downstream grids and the adjacent upstream slack voltages, are received, all agents above the leave agents can execute their local power flow calculation. If the local power flow does not converge, even after several relaxations of the convergence criterion of the NR-Method, the whole DBFS is terminated.

⑧ If the grid state can be determined, the resulting grid load or feed-in is provided by an *PGP* message to all requesting adjacent upstream agents. Voltage-regulated assets are also taken into account accordingly. This process is repeated until the root *GA* is reached.

⑨ The root *GA* then executes a convergence check according to Equation 4.29. As all voltage values have been initialized to 1 p.u. in the first place, in most of the cases, this convergence check will fail after the first iteration. Hence, a second sweep is initiated by the root *GA* again using a *RGP*. All adjacent downstream *GA* now realize that another sweep is triggered and they account for this by sending themselves an *PrepareNextSweepTrigger* ($PNST$). This causes an internal state change and the activation of several routines which prepare another sweep. Afterward, in order to execute an intermediate power flow calculation, an RSV is sent to all adjacent upstream *GAs*.

(10) When all responding PSV are received, an intermediate power flow calculation is executed using the grid asset and adjacent downstream grid power values from the previous sweep and the new voltage values. The remaining part of this sweep is continued in a correspondingly cascading manner again.

(11) The procedure described in the second sweep is repeated until convergence according to Equation 4.29 or the predefined maximum number of sweeps $i_{\text{sweep, max}}$ is reached. Regardless of whether convergence of the overall grid has been achieved or not, the DBFS is terminated then. The termination protocol is initiated by the root *GA* by sending out an *FinishGridSimulation-Trigger* ($FGST$) to all adjacent downstream grids. The receiving *GAs* then also send out $FGST$ to all adjacent downstream grids as well as to all *SPAs* in their grid. Furthermore, in order to announce to the scheduler that they completed the grid simulation, all *GAs* reply to the initial $TM(AST)$ with a *CompletionMessage* (CM) including optional information about the point in time they want to be triggered again. The DBFS from a *GA* perspective is finished with a switch back to its *Idle* state. After receiving all CM the *SimS* knows that the overall system is in sync and hence can advance logical time.

4.5.5 Summary

In this section, one of the core contributions of this thesis, the newly developed DBFS approach, as well as its application within an agent-based simulation model, are presented. The section starts with a general specification of the algorithm requirements in order to be able to execute power flow calculations and grid simulations at scale. Afterward, the mathematical model of the grid, its partioning approach to decompose the overall power flow equations into subproblems, and its features are presented in detail. Furthermore, the specific implementation in a distributed agent-based environment is described in detail, including the actual agent communication and negotiation protocol. The presented approach enables power flow calculations in a concurrent, agent-based model and allows for the execution of large-scale grid simulations on different cluster- or cloud-based computational nodes.

4.6 Photovoltaic Agent

To allow for power system planning, operation and analysis at the distribution level with the help of time series, detailed models of system participants connected to the electricity grid are crucial. PV units, one of the core enablers of the energy system transformation, are almost always connected to the electricity distribution grid. Therefore, their representation within a power system distribution simulation is of high importance. In the following section, the proposed *PvAgent*[11] as well its physical PV unit model is presented in detail.

4.6.1 Motivation for Model Development

In general, most of the *SPAs* in SIMONA constitute of a generic agent hull providing all functionalities required in order to communicate with other agents and the model environment, as well as a physical system model which defines the agent's concrete shape. Therefore, the physical model of a *SPA* is of particular importance. Within this chapter, the main focus is on the development of a new, revised version of a PV unit model. This model can be used within a generic *SPA* in order to represent a *PvAgent* within SIMONA. The development of a new PV model is motivated by the fact that the model presented in [9] is based on several assumptions and simplifications, which produces invalid high power values at small solar irradiance angles in the context of sunrise or sunset (Figure 4.15) which exceed the rated installed capacity of the unit. This invalid model behavior becomes a problem, especially when a simulation is performed for one or more years in which there are days with weather conditions with a small angle of solar irradiance.

The model presented in the following uses a combination of direct irradiance, diffuse irradiance and reflected irradiance to calculate the power feed-in. In contrast to existing models, all radiation components are represented in detail and the number of required simplifications is reduced to a minimum. To the best of the author's knowledge, this combination of different detailed models for the different radiation components has not been carried out in this form yet. As a result of this approach, a detailed PV model is available, which can determine realistic feed-in time series based on real measured or simulated values of a weather station. It is in particular able to calculate realistic power values even for very small irradiance angles. This

[11] By convention, all *SPAs* in SIMONA are named starting with the abbreviation of their physical model followed by the term *agent*. For improved readability, the camel case Notation is used for this purpose.

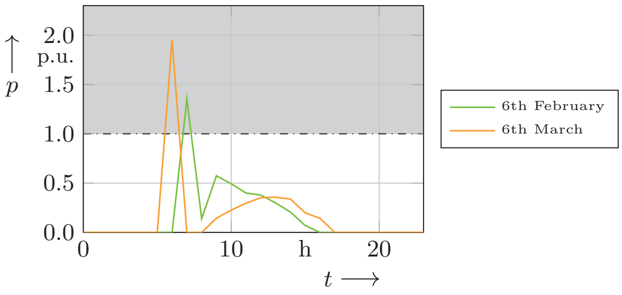


Figure 4.15: Exemplary feed-in profiles of the existing PV model for two days at different sides

functionality allows its stable application in the context of a *SPA* as *PvAgent* for the simulation of the respective power feed-in over several weather years.

In the following, first of all, a short introduction of the generic *SPA* and its interaction protocol is given. Afterward, the newly developed physical model of a PV unit is described in detail. The section ends with a verification and validation of the developed model using different weather input data and a discussion of further options for model quality improvements.

4.6.2 Generic System Participant Agent

The developed *PvAgent* is a generic *SPA* holding a physical model of a PV unit which consequently defines its shape. Although the physical model defines its shape, to integrate the physical model into SIMONA, the behavioral model of the generic *SPA* is crucial. Therefore, to get a full idea about the developed *PvAgent*, knowledge about at least the basic structure of the generic *SPA* is required. The following section briefly describes the necessary foundations of a generic *SPA* including its basic functionalities. For a comprehensive and detailed overview about *SPAs* in general, their setup and interaction protocols, please refer to [11].

Physical Model

The physical model of a generic *SPA* defines its shape. This means, that by holding a PV unit model, the generic *SPA* becomes a *PvAgent* or by holding a private household model, the *SPA* becomes a *PrivateHouseholdAgent* and so on. An overview about all currently available physical models and corresponding agents is given in Table 4.2. Figure 4.16 schematically depicts the general functionality of a physical model in a *SPA*.

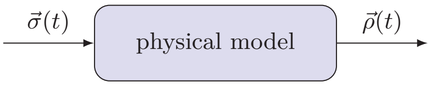


Figure 4.16: Schematic representation of the physical model of a *SPA*

Behavioral Model

While the physical model is agent-specific and defines the shape and physical boundaries of the *SPA*, the behavioral model is generic and hence the same for all different shaped *SPAs*. In order to interact with the overall system, the behavioral model fulfills all requirements of Equation 3.1. The resulting behavioral model, including the agent states and the transition functions, is given in Figure 4.17.

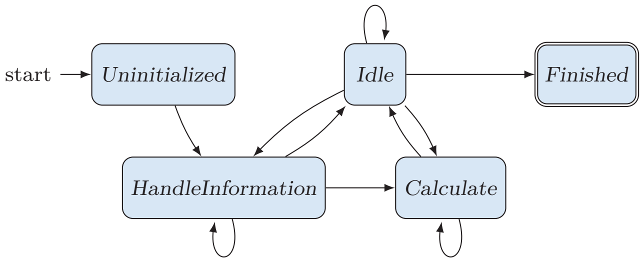


Figure 4.17: Generic *SPA* behavioral model FSM

Every agent starts in the *Uninitialized* state. Within this state, it waits until it receives its model state data, including its physical model. When it has received this data, it processes it in the *HandleInformation* state and switches to *Idle* afterward. Whenever it receives primary or secondary data (see Subsection 4.2.3) a state transition is carried out to *HandleInformation* where the received data is processed. In the case of secondary data, which is the required weather information in the case of a *PvAgent*, the next state transition from *HandleInformation* to *Calculate* is carried out. In the *Calculate* state, the actual physical model calculation, using the previously received secondary data information, is executed. The results of the physical model calculation are then stored in the agent's value store. Following the physical model calculation, the agent switches back to *Idle* and is able to receive and process new messages.

4.6.3 Physical Model of a Photovoltaic Plant

The physical model of the *PvAgent* represents a single PV unit that requires different static parameters to be initialized. While some of them are required for model calculations, other ones are needed for integrating the model into the simulation and the generic *SPA*. The latter include parameters for *reactive power* control[12], the *id* of the asset, its *operation interval* and a *scaling factor* for experiments. However, for the sake of clarity and because they are not required to understand the mathematical model, these additional parameters are neglected in Table 4.6.

Table 4.6: Calculation relevant static input parameters of the PV model

Name	**Variable**	**Unit**
Rated Apparent Power	S_r	kVA
Nominal Power Factor	$\cos(\varphi)$	-
Latitude	ϕ	°
Longitude	λ	°
Surroundings Albedo	ρ	°
Inverter Efficiency Factor	η	%
Azimuth	α_e	°
Module Elevation Angle	γ_e	°

In order to calculate the actual power feed-in of the PV unit, the dynamic weather parameters depicted in Table 4.7 are required.

To calculate the electrical power feed-in according to Equation 4.30, the rated apparent power $S_{\mathrm{r,\,pv}}$ is multiplied with the irradiation yield Y_R, the rated power factor $\cos\varphi$, the inverter efficiency factor η and two correction factors κ_G and κ_ϑ. κ_G is an empirically determined generator correction factor used to take spectral mismatch, partial shadowing of panels, pollution or snow covering into account.

[12]All assets in SIMONA support either $\cos\varphi$-fixed, $\cos\varphi(P)$ or $Q(U)$ reactive power control.

Table 4.7: Required weather parameters of the PV model

Name	**Variable**	**Unit**
Date and Time of Occurrence	-	-
Incident Duration	t	min
Diffuse Solar Irradiance on Horizontal Surface	$E_{\mathrm{e,diff,h}}$	$\mathrm{W/m^2}$
Direct Solar Irradiance on Horizontal Surface	$E_{\mathrm{e,beam,h}}$	$\mathrm{W/m^2}$

κ_ϑ is the month dependant temperature correction factor which allows taking into account panel's temperature.[13]

$$P_{\mathrm{pv}} = -S_{\mathrm{r,\,pv}} \cdot \cos\varphi \cdot Y_{\mathrm{R}} \cdot \eta \cdot \kappa_G \cdot \kappa_\vartheta \tag{4.30}$$

The required irradiation yield Y_{R} is the ratio of total irradiation $E_{\mathrm{pv,\,total}}$ incident on the PV panel surface to total irradiation at standard test conditions $E_{\mathrm{total,\,STC}}$.

$$Y_{\mathrm{R}} = \frac{E_{\mathrm{total,\,pv}}}{E_{\mathrm{total,\,STC}}} \tag{4.31}$$

The total irradiation at standard test conditions $E_{\mathrm{total,\,STC}}$ is the solar irradiance at standard test conditions $E_{\mathrm{e,\,STC}}$ incident on a surface multiplied with the incident duration t, where $E_{\mathrm{e,\,STC}} = 1000\,\mathrm{W/m^2}$.

$$E_{\mathrm{total,\,STC}} = E_{\mathrm{e,\,STC}} \cdot t \tag{4.32}$$

The total irradiation of the, usually sloped, PV panel $E_{\mathrm{pv,\ total}}$ is the sum of beam or direct irradiation $E_{\mathrm{beam,s}}$, diffuse irradiation $E_{\mathrm{diff,s}}$ and reflected irradiation $E_{\mathrm{ref,s}}$ on a sloped surface.

$$E_{\mathrm{total,\,pv}} = E_{\mathrm{beam,\,s}} + E_{\mathrm{diff,\,s}} + E_{\mathrm{ref,\,s}} \tag{4.33}$$

To determine the respective proportions of the beam, diffuse, and reflected irradiation, several calculations which require a basic understanding of the geometric relationships between the sun and a surface relative to the earth at any given time

[13] Both correction factors are already part of the model presented in [9] and are based on long-term measurement analyses presented in [140]. Further information, in particular the underlying lookup tables to determine the correction factors, can be found in [9].

are necessary. For the sake of clarity, the following equations do not contain a comprehensive overview of all required fundamental relationships and angles. However, Appendix B provides all of them in aggregated form.

Beam radiation on sloped surface

With the provided direct irradiance, the beam irradiation on a horizontal surface $E_{\text{beam, h}}$ can be calculated by multiplying the direct irradiance with the incident duration. The derived beam irradiation can then subsequently used to calculate the required beam irradiation on a sloped surface can be calculated using the ratio of the total irradiation on the sloped surface to that on the horizontal surface R_{b}. Equation 4.35 depicts a reasonable approximation based on [141] that can be used to calculate this ratio.

$$E_{\text{beam, s}} = E_{\text{beam, h}} \cdot R_{\text{b}} \tag{4.34}$$

$$R_{\text{b}} \approx R_{\text{b, ave}} = \frac{a}{b} \tag{4.35}$$

where

$$\begin{aligned} a = &(\sin\delta \cdot \sin\phi \cdot \cos\gamma_e - \sin\delta \cdot \cos\phi \cdot \sin\gamma_e \cdot \cos\alpha_e) \cdot (\omega_2 - \omega_1) \\ &+ (\cos\delta \cdot \cos\phi \cdot \cos\gamma_e + \cos\delta \cdot \sin\phi \cdot \sin\gamma_e \cdot \cos\alpha_e) \cdot (\sin\omega_2 - \sin\omega_1) \\ &- (\cos\delta \cdot \sin\gamma_e \cdot \sin\alpha_e) \cdot (\cos\omega_2 - \cos\omega_1) \end{aligned}$$

and

$$b = (\cos\phi \cdot \cos\delta) \cdot (\sin\omega_2 - \sin\omega_1) + (\sin\phi \cdot \sin\delta) \cdot (\omega_2 - \omega_1)$$

where

δ	$=$ sun declination angle
ω_1	$=$ hour angle at request time
ω_2	$= \omega_1 + 15°$ (shifted by one hour)
γ_e	$=$ surface elevation angle
α_e	$=$ sun azimuth
ϕ	$=$ latitude of the surface, here PV unit panel

The necessary parameters for Equation 4.35 can be partly derived from the model's static input parameter set and partly from further calculations. The latitude ϕ of the plant is derived from the corresponding static location input parameter. Assuming that the plant is not re-adjusted by a sun-tracking unit, the azimuthal orientation α_e of the plant and the elevation angle γ_e of the PV plant panel are also available from the static input parameter set. Neglecting the representation of the basic parameter calculation steps, the hour angles ω_1 and ω_2 have to be calculated. The required basic parameter formulas can be found in Appendix B.

In the present model ω_1 corresponds to the actual hour angle ω, for which beam irradiation on a horizontal surface $E_{\text{beam, h}}$ is provided and ω_2 corresponds to the angle shifted by $\Delta\omega = 15\,°$, which is equivalent to one hour. Taking sunrise angle ω_{SR} and sunset angle ω_{SS} into account, ω_1 and ω_2 can be calculated as follows. [14]

$$(\omega_1, \omega_2) = \begin{cases} (\omega_{\text{SR}}, \omega_{\text{SR}} + \Delta\omega), & \text{if } (\omega_{\text{SR}} - \frac{\Delta\omega}{2}) < \omega < \omega_{\text{SR}} \\ (\omega, \omega + \Delta\omega), & \text{if } \omega_{\text{SR}} \leq \omega \leq (\omega_{\text{SS}} - \Delta\omega) \\ (\omega_{\text{SS}} - \Delta\omega, \omega_{\text{SS}}), & \text{if } (\omega_{\text{SS}} - \Delta\omega) < \omega < (\omega_{\text{SS}} - \frac{\Delta\omega}{2}) \end{cases} \tag{4.36}$$

The required angles for sunrise and sunset can be calculated by using Equation 4.37 and Equation 4.38.

$$\omega_{\text{SR}} = \cos^{-1}(-\tan\phi \cdot \tan\delta) \tag{4.37}$$

$$\omega_{\text{SS}} = -\omega_{\text{SR}} \tag{4.38}$$

The consideration of sunrise and sunset as depicted in Equation 4.36 is one of the crucial differences of the presented model compared to other models, in particular, the one presented in [9]. Considering the sunrise and sunset angles when determining ω_1 and ω_2 not only mitigates but avoids high, invalid irradiation peaks resulting in invalid high power feed-in values around sunrise and sunset times.

Diffuse radiation on sloped surface

The diffuse radiation on a sloped surface $E_{\text{diff, s}}$ is calculated using the Perez anisotropic tilt conversion model [142] which divides the diffuse radiation into three parts. First, there is an intensified radiation from the direct vicinity of the sun. Furthermore, Rayleigh scattering, backscatter, which leads to increased intensity on the horizon, and isotropic radiation are considered. The required diffuse irradiation on a horizontal surface $E_{\text{diff, h}}$ can be calculated equivalent to the direct irradiation

[14] For a detailed explanation of sunrise and sunset angles see [141].

on a horizontal surface by multiplying the diffuse irradiance on a horizontal surface with the incident duration.

$$E_{\text{diff, s}} = E_{\text{diff, h}} \cdot \left(\frac{1}{2} \cdot (1 + \cos\gamma_e) \cdot (1 - F_1) + \frac{c}{d} \cdot F_1 + F_2 \cdot \sin\gamma_e\right) \tag{4.39}$$

As can be seen from Equation 4.39, the horizontal diffuse radiation $E_{dif,h}$ is multiplied by a factor, which consists of the horizon brightness index F_1, the sun ambient brightness index F_2 and the ratio of the parameters c and d. The terms c and d are calculated as given below:

$$c = \max(0, \cos\theta_g) \tag{4.40}$$

$$d = \max(0.087, \sin\alpha_s) \tag{4.41}$$

The two indices F_1 and F_2 are determined as shown in equations Equation 4.42 and Equation 4.43.

$$F_1 = F_{11}(x) + F_{12}(x) \cdot \Delta + F_{13}(x) \cdot \theta_z \tag{4.42}$$

$$F_2 = F_{21}(x) + F_{22}(x) \cdot \Delta + F_{23}(x) \cdot \theta_z \tag{4.43}$$

Equation 4.42 and Equation 4.43 require the brightness index Δ, the zenith angle θ_z and six coefficients F_{ij} where $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$. The brightness index can be calculated using the airmass AM, the direct radiation on a horizontal surface $E_{\text{diff, h}}$ and the extraterrestrial radiation I_0.

$$\Delta = AM \cdot \frac{E_{\text{diff, h}}}{I_0} \tag{4.44}$$

In [142], each F_{ij} is selected based on eight empirically predetermined bins for the clearness parameter ϵ. To simplify the lookup process, a method to calculate F_{ij} is to fit polynomials to the coefficients to the ϵ-bin number of the original table. The following polynomial expressions as a function of bin number x with $x \in \{1, 2...8\}$ published in [143] are used for the developed model:

$$F_{11}(x) = -0.0161x^3 + 0.1840x^2 - 0.3806x + 0.2324 \tag{4.45}$$
$$F_{12}(x) = 0.0134x^4 - 0.1938x^3 + 0.8410x^2 - 1.4018x + 1.3579 \tag{4.46}$$
$$F_{13}(x) = 0.0032x^3 - 0.028x^2 - 0.0056x - 0.0385 \tag{4.47}$$
$$F_{21}(x) = -0.0048x^3 + 0.0536x^2 - 0.1049x + 0.0034 \tag{4.48}$$
$$F_{22}(x) = 0.0012x^3 - 0.0067x^2 + 0.0091x - 0.0269 \tag{4.49}$$
$$F_{23}(x) = 0.0052x^3 - 0.0971x^2 + 0.2856x - 0.1389 \tag{4.50}$$

In order to determine the ϵ-bin number x, Equation 4.52 is applied on the result of Equation 4.51 accordingly.

$$\epsilon = \frac{\frac{E_{\text{diff, h}} + E_{\text{dir, h}}}{E_{\text{diff, h}}} + 5.535 \cdot 10^{-6} \cdot \theta_z^3}{1 + 5.535 \cdot 10^{-6} \cdot \theta_z^3} \tag{4.51}$$

$$x = \begin{cases} 1, & \text{if } \epsilon \in [1.000, 1.065] \\ 2, & \text{if } \epsilon \in]1.065, 1.230] \\ 3, & \text{if } \epsilon \in]1.230, 1.500] \\ 4, & \text{if } \epsilon \in]1.500, 1.950] \\ 5, & \text{if } \epsilon \in]1.950, 2.800] \\ 6, & \text{if } \epsilon \in]2.800, 4.500] \\ 7, & \text{if } \epsilon \in]4.500, 6.200] \\ 8, & \text{if } \epsilon \in]6.200, \infty[\end{cases} \tag{4.52}$$

Reflected radiation on sloped surface

The last required part of Equation 4.33, the reflected radiation on the PV panel, can be calculated comparatively simple in contrast to the other radiation components. [144] With the help of the static input parameters albedo ρ, the panel surface slope angle γ_e and the sum of the direct and diffuse horizontal irradiation the reflected radiation $E_{\text{ref, s}}$ can be calculated as follows.

$$E_{\text{ref, s}} = (E_{\text{dir, h}} + E_{\text{diff, h}}) \cdot \frac{\rho}{2} \cdot (1 - \cos \gamma_e) \tag{4.53}$$

4.6.4 Physical Model Verification and Validation

Whenever a model of a real-world system is developed, the verification and validation of its functionality are crucial (cf. Subsection 3.1.1). Doing so does not only ensure that the model's computational results make sense, but it also allows verification that the modeling objective has been achieved. In the following subsection, the developed PV model is first compared against the initial modeling objective to verify its functionality. Afterward, comparison tests are performed to validate the model against an existing, already validated model.

Verification

Recalling Subsection 4.6.1, the main motivation to develop a new PV model was the invalid high power production on hours with small solar irradiance angles as depicted in Figure 4.15. Consequently, it needs to be ensured that the new model does not produce the same invalidly high power feed-in.

For this purpose, the same weather data that led to invalid high feed-in values in the original model is used to calculate the feed-in of the new model. Figure 4.18 shows that the new model no longer produces the invalidly high power values of the original model. Furthermore, the installed nominal power of the plant is no longer exceeded. Hence, it can be concluded that the main objective of the newly developed model, namely to prevent exceeding the installed nominal capacity of the PV model in times of small radiation angles, has been achieved.

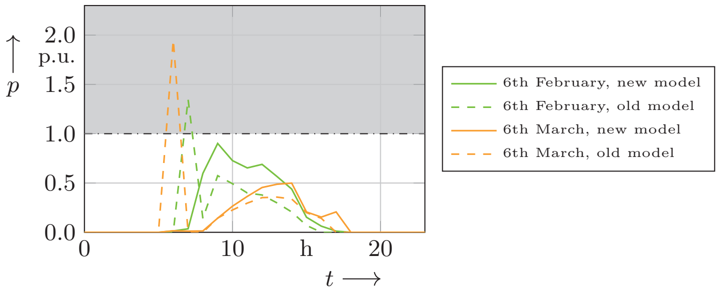


Figure 4.18: Exemplary feed-in profiles of the newly developed PV model compared to the existing one for two days at different sides

Validation

However, while this conclusion may be used as an indicator to verify the expected model behavior, it does say nothing about the actual validity of the calculation results. In order to validate the model, two different approaches are available. The always preferred way would be to compare the developed model with existing field measurement data, that is, solar irradiance and the resulting power production of a PV unit. Another approach is the comparison of the model of interest with another existing model. This approach, also called comparison tests or comparison evaluation, is based on the assumption that an existing model, published and peer-reviewed, can be considered to produce correct results and consequently is a valid representation of the real system it represents. Due to the lack of measurement data, the second approach is chosen to validate the newly developed model.

The comparison evaluation is carried out using a virtual test bench in which the newly developed model and an existing model are confronted with the same weather data and the calculation results of each model are compared against each other. For the result comparison, the calculation results of the model presented in [70] are used along with the corresponding weather data from the MERRA-2 dataset [145] in hourly resolution. Both datasets can be obtained from the Renewables.Ninja (RN) platform for different PV unit configurations and locations. In order to have a significantly large number of samples and consider different weather data from different locations, a sample of $n = 100$ randomly selected locations distributed over Germany have been chosen. An overview of the entire test setup, including all parameters, is given in Appendix C.

Figure 4.19 depicts detailed results of the hourly active power output of an example location during an exemplarily selected week in April, referred to the installed capacity of the models. The figure shows that the output of the newly developed is very close to the output of the RN model. However, although the shape of the plots is close to each other, it is also observable that the developed model results tend to be slightly lower than those of the comparison model.

A more detailed view across all locations separated by season is given in Figure 4.20. The figure shows both models' average hourly power production for different seasons, referred to the unit's installed capacity. The plot supports the assumption already derived from Figure 4.19 of a slightly lower feed-in of the developed model compared to the RN model. While the deviations are relatively small in spring and winter, they are especially noticeable in summer and autumn.

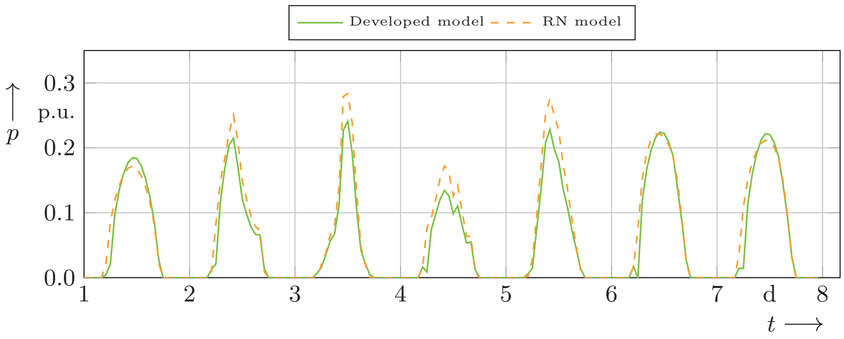


Figure 4.19: Hourly produced active power output for a PV unit located in Germany during an exemplarily selected week in April 2019

Figure 4.21 shows a more systematic investigation of the deviations of the developed model from the comparison model. It plots the hourly Root Mean Square Error (RMSE)[15] per season for all simulations and all units and their locations. All values are again referred to the installed unit capacity. The plot consistently supports the hypothesis of an increased power deviation of the developed model from the comparison model, especially in summer and autumn. Furthermore, the plot also allows the quantification of the model deviations for each season. Even when considering the outliers, with an order of magnitude between 0.0079 and 0.0654 p.u., the deviation is comparatively low. In particular, for the use in the developed simulation model, it is in an acceptable range for most of the intended application cases.

In summary, it can first be stated that the basic function of the developed model meets the requirements and the intended modeling objective. This was demonstrated both by verifying the functionality in the context of a comparison with the old model and by an extensive validation with another existing model.

Nevertheless, it must be critically reflected that deviations occur in the resulting data compared to the existing model. There may be different reasons for this. First of all, as shown in [70], deviations from reality already occur in the reference model. Correspondingly more exact, e.g., more realistic, models therefore inevitably also lead to deviations when comparing the result data. However, it is explicitly impossible to say whether the supposedly more accurate model then matches reality

[15] $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}}$

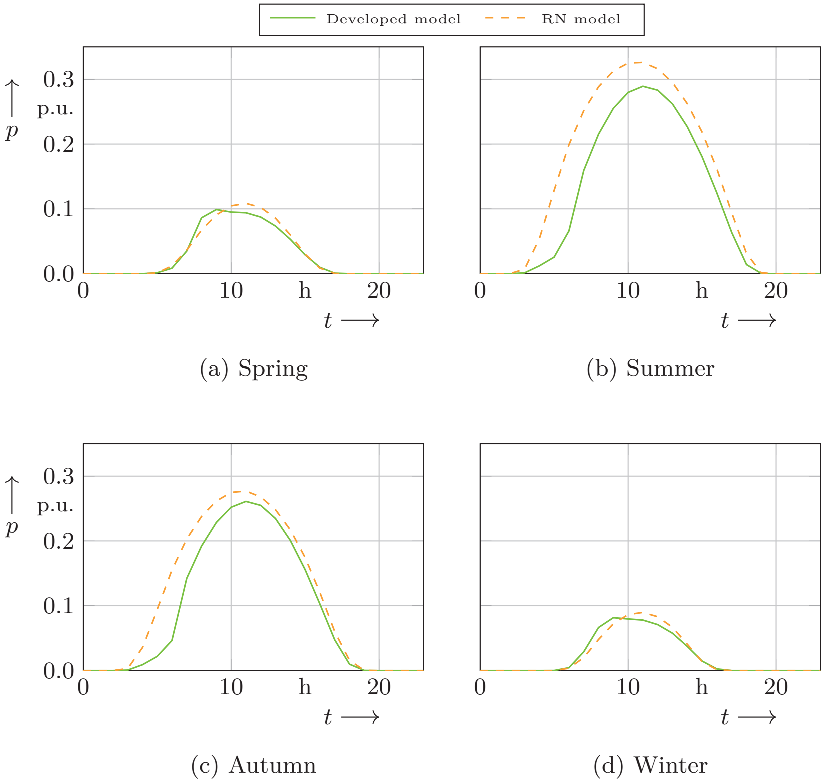


Figure 4.20: Average daily power production comparison between the RN PV model and the developed PV model for $n = 100$ different plants distributed over Germany

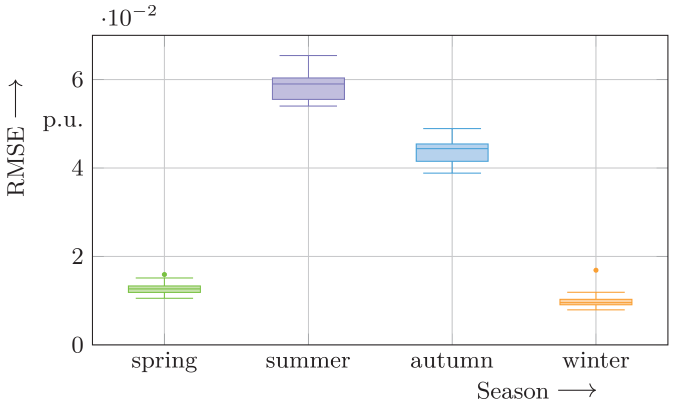


Figure 4.21: Hourly RMSE per season of the RN PV model and the developed PV model for $n = 100$ different plants distributed over Germany

more or less. Furthermore, the parameterization of the RN model is significantly limited in comparison to the developed model. This limitation requires assumptions for some of the parameters used for the parametrization of the developed model, and these assumed parameters may differ from the comparison model. Additionally, the developed model uses different empirically derived correction factors for converter losses and temperature, which may be another source of error that may cause the deviations. Since the verification and validation of the model and not a result approximation as exact as possible to the comparison model is the main target in the context of this thesis, a sensitivity analysis and parameter optimization is omitted at this point.

Therefore, further improvements of the developed model can and should be considered depending on the application case. First of all, it would be desirable to analyze the model results with real measured values, if required and suitable data is available. In addition, further comparative analyses with the model data of the RN platform can be carried out and the parameterization of the developed model can be improved within the scope of sensitivity analyses.

4.6.5 Summary

Motivated by invalidly high power production, which exceeds the installed rated capacity of the existing PV model, a new, more detailed model is presented in this section. The new model considers direct, diffuse and reflected irradiance for the calculation of the resulting power production. After outlining the motivation, the functionality of a generic *SPA* is shortly described, followed by an extensive formal description of the developed physical model, which makes a generic *SPA* a specific *PvAgent*. After the formal description of the physical model, its functionality is verified and validated. First, it is compared with the old model to ensure that the invalidly high power production does not occur anymore. Afterward, extensive comparison validations against another established model are carried out and the results are discussed. The verification and validation results allow the conclusion that the model works as expected and can be used for the intended application cases within the agent-based environment. However, as always in model development, different options are available to improve the model quality, which are discussed accordingly and may be considered for further research activities.

4.7 Evaluation of the Developed Overall Model

Section 4.1 defines the main specifications of the developed simulation model, which are the leading guidelines for the development of the model. The following section shortly evaluates the specifications' degree of fulfillment, summarized in Table 4.8.

Section 4.3 provides an *extensive description of the notion of an agent in* SIMONA, including a formalism for the three currently available agent types. The discrete-event simulation model presented in Section 4.4 enables the simulation of *flexible input- and output-data time step lengths*. The simulation can combine time step lengths of a few seconds up to several hours or days in a single simulation, while the smallest time step length is 1 s. The new data flow concept of SIMONA allows the *consideration of pre-calculated exogenous primary or secondary data*. Data sources can be databases, file sources or external co-simulations. The model application programming interfaces further provide different options to *externally control the simulation*, e.g., by co-simulation frameworks like *mosaik*. Disabling the DBFS power flow also allows a *standalone system participant simulation*.

While it is ascertainable that, *in general*, all system participant specifications are fulfilled, a detailed examination of individual aspects reveals some minor limitations. Section 4.6 describes the *detailed bottom-up models* with a focus on single

Table 4.8: Evaluation of the defined specifications of the developed simulation model

General Model Specification	
Description of the notion of an agent	✓
Flexible input and output data time step length	✓
Partial up to full consideration of pre-calculated exogenous data	✓
Controllability by and consideration of data of co-simulations	✓
Standalone system participant simulation	✓
System Participants	
Detailed bottom-up models with focus on single entities	✓
Consideration of entity interdependencies	✓
Consideration of individual behavioral aspects	✓
Clear separation between behavioral and phyiscal model	✓
Validity of the physical model	(✓)
Grid Representation and Power Flow Execution	
Validity of the physical model	(✓)
Exchangeability of the power flow algorithm	✓
Concurrency by design for scalability	(✓)

entities using the example of a *PvAgent*. This section also outlines the *clear separation between behavioral and physical models of SPAs* in SIMONA. The comparably simple replacement of physical models is demonstrated by a substitution of an existing physical PV model with a new one. Subsection 4.6.4 presented the *verification and validation* of the developed physical PV model. However, a general statement concerning verification and validation of *all* physical models available in SIMONA cannot be made based on the exemplary PV model validation. Therefore, the fulfillment of this specification is limited to the PV model. In principle, however, all models in SIMONA are checked accordingly for verified and validated functionality. However, this aspect requires further research. Section 4.6 and Subsection 4.5.3 outline the considered *individual entity interdependencies and behavioral aspects* of the currently available agents in SIMONA. Supplementary important aspects to the behavioral and interaction model of the *SPA* can be found in [11].

In Section 4.5, the newly developed DBFS power flow algorithm is presented. The algorithm meets the *specification of exchangeability* twice. First, the used NR-Method for single subgrid calculations is replaceable by any other power flow algorithm. Second, it is even possible to fully replace the DBFS including its communication protocol. However, this requires the development of a new communication protocol and therefore goes along with high effort. The evaluation concerning *concurrency by design* is somehow ambivalent. On the one hand, the developed partioning approach and its distributability on several agents *in general* allow for concurrent execution. On the other hand, if the agents are not implemented to execute them concurrently, the algorithm design also allows a sequential execution. Therefore, the specification is only partially met. The same is true for the *physical grid model validity*. While the DBFS algorithm and the underlying NR-Method provide plausible results, a specific validation is not done yet. However, a detailed evaluation and validation of the transformer model used can be found in [11]. Additionally, Chapter 5 also partially validates SIMONA's power flow and therefore implicitly contributes to the validation of the physical grid model.

In summary, it can be stated that the majority of the model specifications formulated at the beginning of the model development can be fulfilled both theoretically and practically. This is taken up again in Chapter 6 where an outlook on further research and development possibilities of the model is given.

5 Application Example

While the previous chapter focuses on detailed descriptions of several core functionalities of the developed simulation model, subsequent sections focus on its application for different investigations.

The first investigation contributes to the verification and validation of the developed Distributed Backward-Forward Sweep Algorithm (DBFS) power flow algorithm. It compares available SimBench reference voltage profiles at the nominal operating point with the corresponding simulation results. The second application example is a fictional Photovoltaic (PV) park connection request with multiple expansion stages. There, SIMONA is used to simulate and evaluate the impact of the PV park on the electricity grid, including the consideration of flexibility measures.

Both application cases make use of SIMONA-Ecosystem tools and SimBench benchmark grid models. The former are tools that were developed in the context of the research for SIMONA in order to support pre- and post-processing steps in power system analyses. The latter are freely available benchmark grid models that also contain reference results and geographic information in addition to the grid structure. A short introduction on both can be found in Appendix D.1 and Appendix D.2.

5.1 Power Flow Validation at Nominal Operating Point

This section validates the DBFS power flow calculations in SIMONA using SimBench benchmark grid models. In addition, this section also contributes to the analysis of the used grid models' electrical properties and therefore complements existing work such as, e.g., [146]. Since the power flow validation is crucial for the developed DBFS and also the transformer models, the subsequent investigation has been carried out jointly with a research partner and the results are published in [11] as well.

As noted in Appendix D.2, every SimBench grid contains node voltage magnitude and angle reference profiles, usable to detect deviations and validate a power flow algorithm. Thereby, each of these profiles represents the unscaled nominal operating point without any voltage setpoint control. [147]

The data set used for the following validation consists of 78 selected SimBench benchmark models. Selected based on their relevance for SIMONA, they comprise 18 low voltage grids, 12 medium voltage grids with equivalent surrogate low voltage grids and 48 combined level grids. The combined grids consist of the medium and the low voltage level with at least one low voltage grid explicitly modeled. Table D.1 gives an overview of the SimBench codes of all used grids.

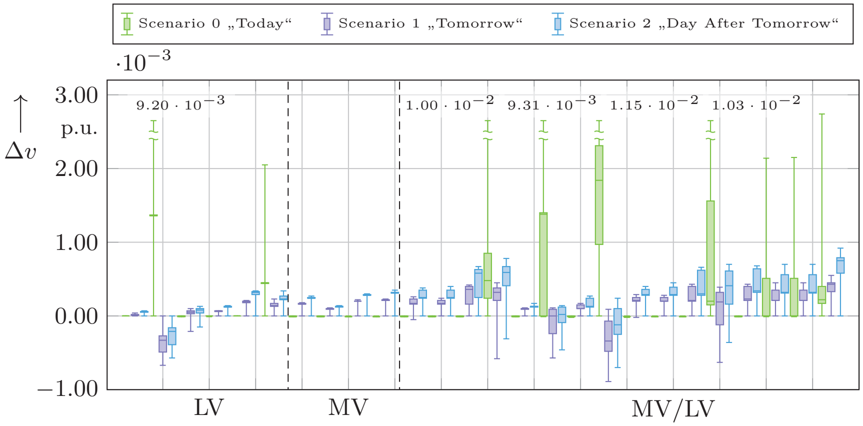
The validation process does not only include SIMONA, but also parts of the SIMONA-Ecosystem, namely simbench2psdm and PowerSystemDatamodel (PSDM), and therefore validates not only the simulation results but also the interplay of the applied tools. Before the actual validation can take place, the simbench2psdm converts every grid model from the SimBench data format into PSDM data format. Afterward, SIMONA executes simulations at the nominal grid operating point for each of the selected 78 grids. The simulation results are subsequently compared with the corresponding node voltage magnitude and angle reference profiles.

Figure 5.1 shows the statistical evaluation of the deviations between SIMONA simulation results and SimBench reference profiles at the nominal operating point. The Interquantile Range (IQR) spans the 25 % to 75 % quantiles. The whiskers in Figure 5.1a span the maximum and minimum occurring values, while in Figure 5.1b they span the 2.5 % and 97.5 % quantiles and outliers are dotted. Positive values indicate a larger value in the simulation results compared to the reference profiles. Each grid forms a group for the three scenarios, “0 - Today”, “1 - Tomorrow”, “2 - Day After Tomorrow”, while the color indicates the specific one.

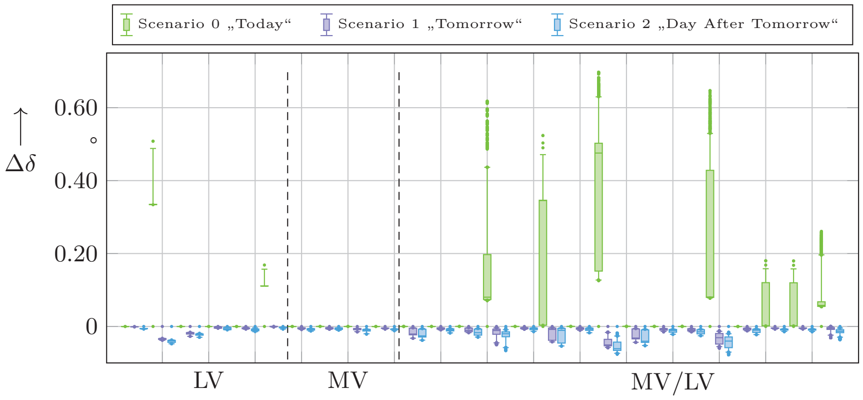
The node voltage magnitude comparison shows a small to moderate deviation between the simulation results and the reference profiles (Figure 5.1a). For scenario 0, the largest share of the maximum deviations in terms of the amount is $10 \cdot 10^{-6}$ p.u. and the largest absolute overall deviation is $1.15 \cdot 10^{-2}$ p.u.. More significant but still moderate deviations for this scenario can be found primarily in the coupled simulations. In this context, it is crucial to consider that the reference profiles only specify values with an accuracy of five decimal places, limiting the comparison to a corresponding precision.

Analyzing the general development of the deviations further reveals additional aspects. First of all, except for a few grids, the absolute deviation values are almost exclusively positive. This observation concludes that the SIMONA’s simulation results tend to be above the reference values and may overestimate the voltage magnitude. A second observation is a general trend towards higher deviations with an increasing scenario number.

Complementary to the node voltage magnitude analysis, Figure 5.1b shows the statistical evaluation of the node voltage angle deviations. Comparing the node angle deviations with the respective node voltage magnitude deviations shows that almost the same phenomena occur. For the majority of the scenario 0 values, the



(a) Node voltage magnitudes



(b) Node voltage angles

Figure 5.1: Statistical evaluation of the deviations between SIMONA simulation results and SimBench reference profiles at the nominal operating point for 78 benchmark grids

angle deviations are almost zero. Furthermore, the deviations also increase with rising scenario numbers, and the highest number of deviations also occurs in the coupled simulations. In many cases, negative angle deviations correspond to positive deviations of the magnitude. Exceptions on this are the higher positive angle deviations in the coupled simulations.

Conclusion

Based on the presented results of the performed delta analysis, it can be stated, first of all, that the power flow in SIMONA can be assumed to be verified and validated. In particular, the deviations in the separate grid simulations of low and medium voltage are minimal regarding the nodal voltage magnitude and the voltage angle. Most of the deviations, as well as the largest in terms of their absolute value, occur in the simulation results of the coupled medium and low voltage grids. Although there are still some grids, especially in scenario 0', which show only a marginal to no deviation, all in all, the share of deviations in the range up to absolute $1 \cdot 10^{-3}$ is noticeably higher than in the separate simulation runs. In total, even the maximum deviations with $1.15 \cdot 10^{-2}$ p.u. for the node voltage magnitude and $0.629, 79°$ for the node voltage angle are in an acceptable range for the application cases of SIMONA. For the sake of completeness, Table D.2 gives an aggregated overview of the upper and lower voltage magnitude and angle deviations.

Regarding the deviation's causes, different explanations are possible. First, the increased deviations in the coupled simulation runs are particularly noticeable. This observation indicates that SIMONA handles multi-voltage level grids differently than the simulation used to calculate the SimBench reference results. Therefore, the DBFS, as well as SIMONA's transformer model, may be one starting point for further investigations of the deviations in coupled simulations. Another explanation that is also valid for the separated low-voltage and medium-voltage simulation could be corrupt reference data. As already stated earlier, the SimBench data sets are partially incomplete and therefore, it cannot be fully guaranteed that the provided data is correct. A different explanation for this phenomenon may be the somewhat imprecise wording for the future scenarios in [147]. Some aspects of the concrete parameterization of the nominal operating point in scenarios 1 and 2 are not explained in detail, which may lead to invalid assumptions in grid data conversion. These invalid assumptions could lead to reduced power consumption of system participants, e.g., loads or batteries, in SIMONA compared to the reference result data. In the context of this thesis, however, the main reason for these positive voltage magnitude deviations combined with positive angle deviations could not be determined, requiring further investigations on the exact causes.

In conclusion, however, all deviations are within an acceptable range and sufficient for the use cases of SIMONA. Considering that the SimBench reference results' precision is limited to five decimal places and SIMONA also achieves this precision in the majority of the investigated cases, even more justifies this conclusion. Independently of this, however, future research work can and should put effort into increasing the numerical accuracy in order to improve the quality of the results further.

5.2 Use Case: Photovoltaic Park Connection Request

This application example investigates a common question from power distribution system planning - a connection request for a new PV park with a fixed first and an optional second expansion stage. In this context, SIMONA simulates a multi-voltage level benchmark grid including all connected system participants to derive utilization time series for the base case, the initial PV park stage, and the expansion stage. The section further demonstrates the developed model's capabilities to simulate flexibility by investigating the impact of a $Q(V)$ voltage setpoint control for the expansion stage. The simulations combine external primary data with internal model calculations highlighting another newly developed feature of SIMONA.

5.2.1 Defined Supply Task and Grid Specification

The basis of the use case is the SimBench benchmark grid with the grid code "1-MVLV-rural-4.101-0-no_sw" which mimics a combined medium-low-voltage grid in a rural area. Its scenario "0" corresponds to the current state of a comparable real-world grid and consists of a detailed medium voltage level grid topology with one explicitly modeled low voltage grid. Equivalent surrogate models represent additionally connected low voltage grids. This application case utilizes the available grid topology without switches. All SimBench grids contain rudimentary geographic information. The selected one's reference node is located at a geographical position with latitude 53.64° and longitude 11.40°. Figure 5.2 shows the resulting schematic grid topology neglecting connected system participants and low voltage surrogate models. To demonstrate SIMONA's capabilities of hybrid simulations, that is, combining externally provided primary data and internal model calculations, all simulation runs utilize the SimBench system participant time series and combine them with internal model calculations of the PV park.

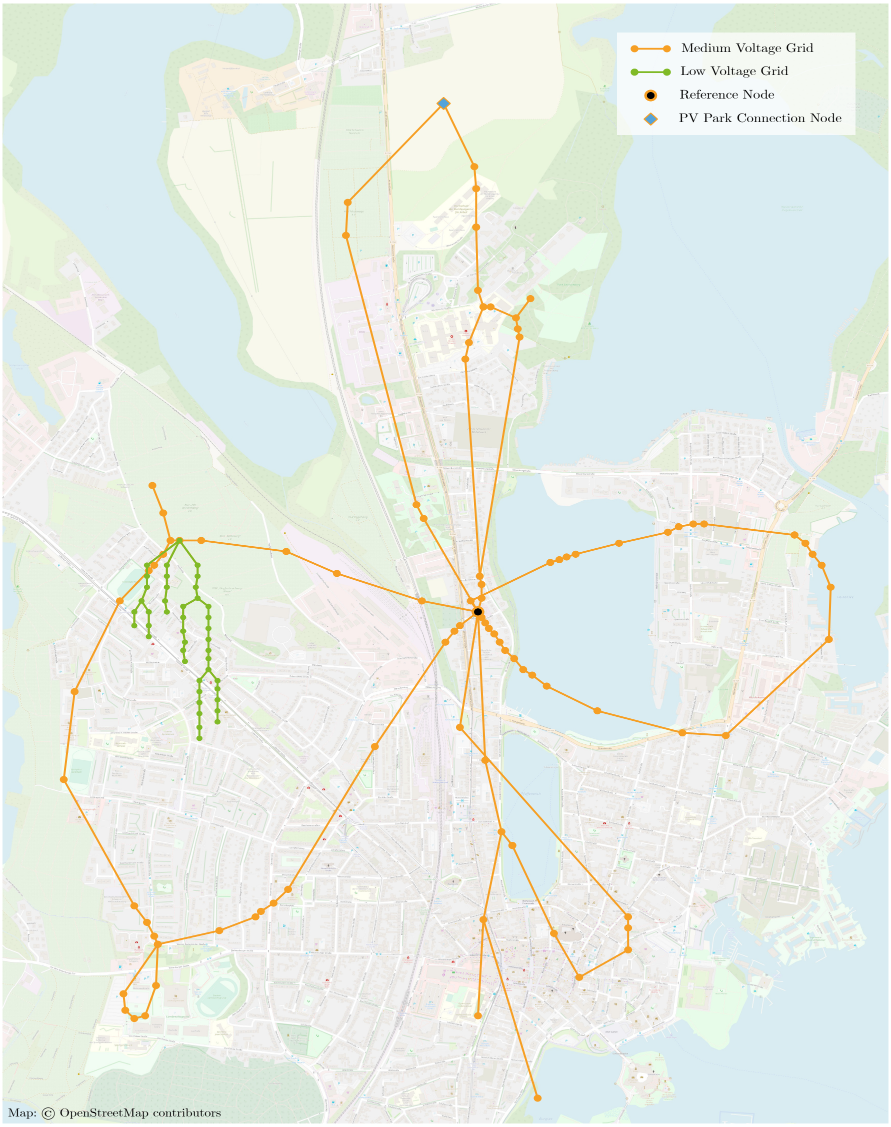


Figure 5.2: Schematic overview of the used SimBench grid model

The medium voltage level grid topology at 20 kV is a partially meshed, open ring system consisting of 102 nodes which are served by two 25 MVA transformers from the 110 kV adjacent superior grid. The connected detailed low voltage grid at 0.4 kV consists of three feeders with a total of 43 nodes and is supplied by a single 0.4 MVA transformer to the medium voltage level.

The ratio between load and feed-in in the unmodified base case corresponds to an installed capacity of 18,554.83 kW to 25,564.98 kW, or 42.05 % and 57.97 %, respectively, based on the absolute sum of installed load and feed-in. Table D.3 in the appendix gives an aggregated overview of all important technical parameters. The described grid specification and supply task represent the application's *base case*.

5.2.2 Future Supply Task Definition

Starting from the *base case* this application example investigates the impact of a connection request of a PV park with two expansion stages. Figure 5.2 marks the connection node in the northern part of the grid where the park should be connected. The request consists of two construction stages, while the first one is mandatory and the second one is optional. The rated apparent power of each stage is 2000 kW, such that the second expansions stage would have an installed feed-in capacity of 4000 kW.

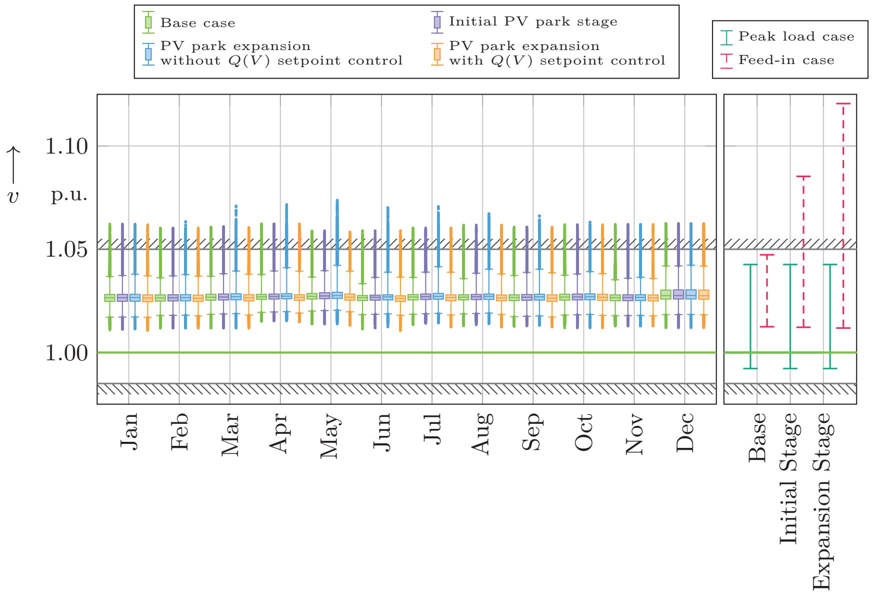
It is assumed that the entire park uses the same hardware in both expansion stages. All park panels align in the same direction and other parameters such as power factor, inverter efficiency or azimuth are the same for the entire park over both expansion stages. Solar tracking is not available and therefore, not considered in the simulations. The corresponding parameters are given in Table D.4 in the appendix. In addition to these properties, the second expansion stage of the PV park is simulated not only with a fixed power factor but also with a $Q(V)$ setpoint control to investigate its impact on the node voltages. Its operating characteristic as given in Figure D.1 in the appendix.

All existing connected system participants and their operation mode remain the same as in the *base* case for the initial PV park stage, as well as both investigated cases in the expansion stage.

5.2.3 Simulation Results

This section describes and analyzes the different simulation results. All time series results comprise simulations of the full year 2016 in hourly resolution. The PV park simulates its power feed-in using measured weather data from 2016 from the German Meteorological Service (DWD) provided by SIMONA's internal *WeatherService*. Grid protection and short-circuit calculations are not part of SIMONA and are therefore not considered in the context of this application case. To allow for a comparison with the conventional planning approach, in addition to SIMONA's time series simulations, the application case contains peak load and feed-in case power flow calculations. The operating point in the peak load case is composed of 100 % of load and 0 % feed-in from distributed energy resourcess (DERs) and the feed-in case is composed of 15 % of load and 100 % of feed-in from DERs.

Figure 5.3 shows an aggregated overview of all simulation results. It depicts the node voltage magnitude distribution in the medium voltage grid for all SIMONA time series simulations (Figure 5.3a) and the conventional power flow calculation results (Figure 5.3b).



(a) Simulated time series (b) Conventional

Figure 5.3: Medium voltage level node voltage magnitude distribution for all investigated cases

In Figure 5.3, the reference voltage of 1.0 p.u. is indicated by a solid green line. The gray shaded boundaries correspond to the node voltage limits for the medium voltage level, assuming a fixed voltage band split according to [148].

In the boxplots of Figure 5.3a, the IQR, the “box”, spans the 50 %-quantile and the whiskers span the 95 %-quantile. Dots above or below the end of the whiskers indicating outliers. The dash inside the IQR denotes the median of all values. The results of the time series simulations are grouped for each month and color-coded per simulated case.

The whiskers in Figure 5.3b span the area of the highest and lowest voltage magnitude occurring in the grid for each peak load and feed-in case.

Starting with the conventional analysis given in Figure 5.3b, it is observable that the grid operates within the defined voltage band split boundaries for the peak load and the feed-in case in the base case. Although the feed-in case is only slightly below the upper border, no operation limit violation occurs. Furthermore, it is noticeable that the difference between the lowest and highest voltage magnitude value is larger in the load case than in the feed-in case. The reason for this is that the load case does not consider feed-in from DERs and therefore the reactive power consumption of the loads is dominant.

This situation changes for the initial PV park stage and the expansion stage. Due to the now connected park, the differences between the minimum and maximum voltage magnitude values in the feed-in case become larger than in the load case. In contrast, the load case differences stay the same for all cases. Both observations are logical since the newly connected PV park does not influence the load case but only the feed-in case. However, the initial and the expansion stage in the feed-in case now violate the defined voltage boundaries. In the conventional planning process, this situation would trigger grid optimization measures followed by required grid reinforcement or expansion.

Taking a look at the simulated time series results in Figure 5.3a, first of all, it shows that the grid is in general operated above a voltage magnitude of 1.0 p.u. for all nodes in the medium voltage level. This fact applies to the base case as well as to any of the PV park stages. Furthermore, it shows that the defined voltage boundaries are violated during some time of the year, even in the base case. While 95 % of the voltage magnitude values in each month are below the defined boundaries, they are exceeded several times over the year, as shown by the outliers. This exceedance occurs for the base case as well as for the PV stages.

Comparing the time series simulation results with the conventional feed-in case calculations, it is noticeable that the feed-in situation in the conventional approach underestimates the node voltage in the base case and significantly overestimates the node voltage magnitudes for the connected PV park. Additionally, in contrast to the conventional approach, the time series-based simulation shows that only in 2.5 % of the simulation results per month, an exceeding of the specified voltage band occurs for the base case and all PV park stages. In 95 % of the monthly results, no violation of the voltage band occurs. Thus, this does not lead to a breach DIN EN 50160 either, requiring the values to be within the voltage band in 95 % of the annual 10 min averages. Additionally, while the initial PV park stage indicates a significant impact on the voltage magnitude in the conventional feed-in case, the time series simulation reveals that its effect is only marginally. A possible explanation for this is that in the feed-in case 100 % DER feed-in is assumed, which rarely or never occurs in reality.

While in the conventional case, measures would have to be taken both for the initial phase of the PV park and its expansion stage, the time-series-based approach allows further analysis of whether and, if yes, which measures are required. These measures distinguish between those that can counteract permanent and long-lasting voltage band violations, e.g., electric batteries, and those that can compensate temporarily and infrequently occurring voltage band violations. These include, for example, the installation of a $Q(V)$ setpoint control as a flexibility option. While this is not suitable for addressing permanent operating limit violations, it allows for resolving sporadic voltage band violation issues.

Taking into account the above analysis, a $Q(V)$ setpoint control is of interest to the present case. And even if no violation of DIN EN 50160 occurs in any of the time series-based simulation cases, the outliers have a significant level, especially in the second expansion stage. These high magnitudes may be acceptable for a small number of times during a year, but a Distribution Grid Operator (DSO) would possibly formulate specific connection requirements regarding the reactive power characteristics of the PV park. Accordingly, another investigation of the expansion stage with a $Q(V)$ setpoint control is carried out.

The results of this investigation, also shown in Figure 5.3a, apparently show the influence of the control. The outliers are significantly lower in magnitude than in the simulation of the expansion stage without $Q(V)$ setpoint control. Moreover, in some months, e.g., March, the IQR even is below that of the first expansion stage, indicating a positive impact of the control unit in general and not only for the additionally installed feed-in power.

Figure 5.4 takes a closer look at the differences in the reactive power injection between the expansion stage without and with the setpoint control. All values are referred to the rated reactive power of the PV park.

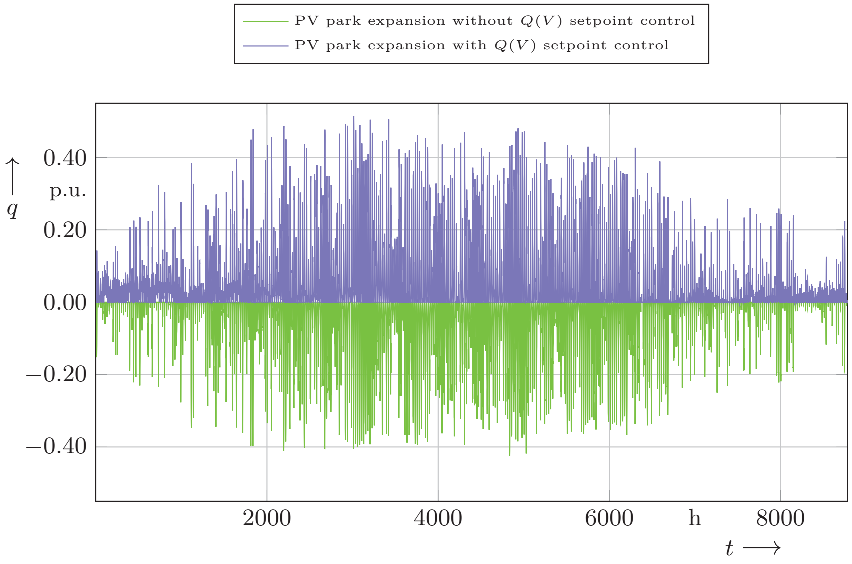


Figure 5.4: Reactive power feed-in of the simulated PV park without and with $Q(V)$ setpoint control

As shown in Figure 5.4, the PV park produces reactive power when simulated with a fixed power factor, which leads to increased node voltages as also observable in Figure 5.3a. The utilization of the reactive power control reverses this behavior, and the PV park now consumes reactive power, which reduces the node voltages accordingly. It is also noticeable that the reactive power consumption tends to be slightly higher than the reactive power provision without the reactive power control. Thus, compared to simulation with fixed power factor, more inductive reactive power is available in total at the same time point.

A comparison of the node voltage magnitudes at the connection node of the PV park, shown in Figure 5.5, confirms this observation.

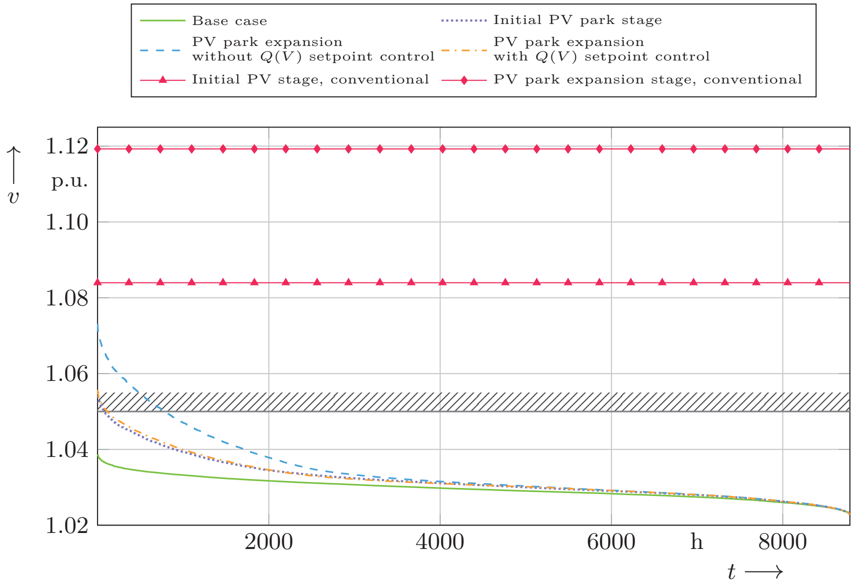


Figure 5.5: Node voltage magnitude ordered duration curve for the PV park connection node

Figure 5.5 shows that the use of a $Q(V)$ control in the second expansion stage case significantly reduces both the frequency and the magnitude of the voltage band violations. Furthermore, it again confirms the observation already visible in Figure 5.3a regarding the fact that almost similar voltage magnitude values as in the initial stage are reached. The selected representation also shows that for the relevant connection node of the PV park, the number of voltage band violations is in the range of a few hours per year. Comparing these realistically occurring violations with the results of the conventional feed-in cases is another indicator of the overestimation of the actual magnitude and frequency of voltage band violations in the traditional planning procedure. This situation may cause an over dimensioning of the grid and subsequently to higher, undesirable capital commitment costs.

5.2.4 Conclusion

In the preceding section, SIMONA is applied to a benchmark grid to investigate a grid planning question. The section exemplarily demonstrates its different capabilities. First of all, its abilities to simulate different grid scenarios are shown. Secondly, the simulations further demonstrate two different execution modes. The first one only uses externally provided primary data for the system participants, while the second combines externally provided primary data with internal model calculations in a hybrid simulation. Finally, the application case demonstrates SIMONA's capabilities to consider flexibility in the simulations using an exemplary $Q(V)$ setpoint control.

These demonstrations allow the conclusion that an extensive, time series-based grid analysis is possible with the developed model. Furthermore, the simulation data is usable for grid analyses and planning purposes. Even though the application case is comparably simple, the results presented here already indicate the potential of time series-based planning compared to conventional planning. In particular, the use case shows that conventional grid planning is not able to realistically estimate the occurring feed-in of DERs, which can lead to oversizing of the grid. The extensive data, resulting from time series-based, behavior-oriented grid simulations, enables comprehensive evaluations and analyses of the investigated grid, exceeding today's procedure in the context of existing grid planning approaches, even innovative ones, such as peak capping or similar.

The resulting data is suitable to perform statistical evaluations such as probability analyses. Additionally, available information can further enable improved risk assessment and evaluation of necessary or postponable investment measures. This information also opens up the possibility of integrating short-term measures for temporary remediation into the planning process. An example of this is the demonstrated $Q(V)$ set point control used in the second extension of the PV park to improve the voltage band. Furthermore, a detailed evaluation of temporal correlations allows the evaluation of alternative planning options, such as the installation of electrical storage systems and the comparison of their investment costs with the costs of conventional grid expansion.[16] In connection with multi-voltage level simulations and the derived observations, effects on lower or higher voltage levels can be detected and addressed.

With the increasing convergence of grid planning and operation, SIMONA is modeled as a simulation service that is capable of addressing questions concerning power

[16] Assuming that the regulation is going to allow the operation of batteries by DSOs or that contracts with battery operators for grid support exist.

system analysis, grid planning and operation. The system can be used to address questions regarding *Redispatch2.0*, *Redispatch3.0* or distribution grid state estimation. Chapter 6 gives a short overview on ongoing research projects dealing grid operation questions, which already apply SIMONA for their investigations.

5.3 Summary

This chapter demonstrates the capabilities of the developed agent-based simulation model SIMONA and validates its functionality using two different use cases.

After some preliminary remarks, 78 SimBench benchmark grid models are simulated with all assets operated at their nominal operation point. The resulting voltage magnitudes and angles are subsequently compared with the provided SimBench reference results to validate the proper functioning of SIMONA. In most cases, the results show only marginal deviations from the reference results. Thus, the functionality of SIMONA is considered to be verified and validated. Regarding the deviations occurring in some grids, further research should investigate their origins and may adapt SIMONA to improve the quality of the simulation results further.

In the second application case, SIMONA is used to investigate a grid planning question. A SimBench benchmark grid is used again to which a new PV power plant park with two subsequent expansion stages is connected. The application case demonstrates several essential capabilities of SIMONA, including hybrid simulations of external primary data and internal model calculations and the simulative consideration of flexibility through the example of a $Q(V)$ setpoint control.

This chapter can be summarized by stating that the validation of the model proves its proper functioning. In addition, a possible application for the simulation model is presented and selected analyses are carried out. The application case is part of a series of other already published work in the field of grid planning, operation and analysis in which SIMONA is involved. For example, in [P4], grid operation simulations are carried out and the authors investigate the influence of artificial intelligence-based control algorithms to integrate charging stations for Electric Vehicles (EVs). [P7] demonstrates the applicability of the developed simulation model on a grid model with more than 50,000 nodes across more than four different voltage levels. [P5] uses the results of the grid simulation for automated grid expansion planning combined with a genetic algorithm. An outlook on further applications and currently ongoing research projects utilizing SIMONA for different purposes is given in Chapter 6.

6 Summary and Outlook

This thesis conceptually designs, models, and implements a flexible power distribution system model for time series generation using discrete-event modeling and simulation techniques combined with an agent-based simulation model. The following chapter summarizes the main findings, critically discusses the used approaches and gives an outlook on further research questions.

Summary and Contributions

The transformation of the energy system is permanently changing the planning and supply task of electrical distribution grids. Increasing shares of distributed energy resources (DER) as well as increasing interdependencies between electricity, gas, heat, and mobility sectors pose a variety of new challenges for Distribution Grid Operators (DSOs). These include increasingly bidirectional power flows between transmission and distribution grids and a subsequently increased communication and coordination effort, a growing number of small and micro power plants, or newly emerging concepts for *smart grid* or *smart market* applications. For this reason, previous grid analysis, planning, and operation conventions have become obsolete. Nevertheless, to ensure a stable and secure electricity supply, leverage new potentials, and reduce risks, innovative, flexible modeling and simulation approaches for the analysis of the emerging overall system are required. These models must accurately represent the existing power distribution system as well as potential future systems. Furthermore, simulated system states must correctly consider spatial and temporal dependencies and influences from other sectors. This can be achieved through generation and utilization time series of the electricity grid, all connected system participants as well as their individual behavior. If available, these time series can support the planning and operation of modern, active power distribution systems.

This thesis focuses on the conceptual design, modeling and implementation of a flexible model for power distribution system simulation to generate power system time series. With this being the main objective, the research question is:

How to model and implement an efficient and flexible distribution power system simulation with a focus on the electricity grid that is capable of serving as an analysis and development testbed to generate time series for modern planning, operation and analysis purposes?

In the course of answering the research question, this thesis builds on and continues existing research. Its main findings can be summarized as follows.

Based on a detailed analysis of the challenges of the energy system transformation and their influence on the planning and operation of power distribution grids, three central components for modeling and simulating the sector-coupled power distribution system can be identified. These components include steady-state power flow calculations, system participant simulations and support for co-simulation frameworks to represent coupled sectors. Based on these findings, the functionality of a large number of existing approaches and tools is analyzed in the context of a literature review. It turns out that none of the presented tools alone meets the requirements of a simulation tool for modern, active distribution grids. The reasons for this are manifold, but especially the lack of consideration of individual system participant behavior, the unavailability of system participant models in general, and a lack of scalability, e.g., their applicability on large grids, are common causes. Furthermore, the proprietary character of some tools represents a partially insurmountable barrier regarding holistic, sector-coupled simulations.

Due to these findings, a new simulation model, SIMONA, aiming to solve current issues in power distribution system analysis, planning and operation, is developed in cooperation with a research partner. The resulting overall concept is depicted in Figure 1.3. The conceptual design of such a model on the one hand, and the possibility of its efficient and scalable application on the other hand, however, involves several challenges. These include the high number of system participants in the power distribution system, the consideration of their individual behavior and interactions, and the necessary reflection of other, coupled sectors. Given these challenges and existing research findings, an agent-based modeling approach is combined with a discrete-event simulation approach to form a unified simulation model. This combination allows the modular construction of autonomously acting agents with different degrees of intelligence and their efficient simulation execution due to the representation of temporal behavior by a discrete-event approach.

Especially with regard to the structure of the distribution grid, the decomposition concept of Agent-based Models (ABMs) and Agent-based Simulation (ABS) approaches facilitates the bottom-up modeling of a large number of individual power plants and is therefore particularly well suited for modeling and simulation of the power distribution system. However, the very rough definition of an agent first requires a clear description of which entities in a modeled system constitute an agent and which do not. This work contributes to this by introducing the three agent types in SIMONA and a general classification of SIMONA agents. In this context, the foundations for the clear separation of an agent's behavioral logic and the restrictions of its physical model are presented. Particular emphasis is put on a detailed description of the resulting modular concept of SIMONA, which ensures the comparatively simple exchange of physical models while preserving the existing interaction behavior and protocol.

SIMONA's agent model consists of two agent entities, the *SystemParticipantAgent* (*SPA*) and the *GridAgent* (*GA*). The generic *SPA* [11] model, built on the foundations of separated behavioral logic and physical unit model, enables the detailed representation of any load, Wind Energy Converter (WEC), Photovoltaic (PV) system or other system participants as an agent. In parallel, additional individual behavioral interactions can be added on a participant-specific basis. These include technical behaviors such as using different reactive power control characteristics to provide grid flexibility or, for example, market- or grid-oriented operation modes. The second agent entity, the *GA*, is the counterpart to the *SPA*. While a *SPA* is responsible for a physical model of a particular system participant, the physical model of the *GA* consists of a electricity grid model, its topology and associated nodes, lines and transformers. It is responsible for performing complex power flow calculations to determine node voltages, grid and asset utilization. In addition, it may consider innovative grid assets such as transformer tapping or flexibility restrictions, e.g., curtailment negotiations with specific *SPAs*, in the course of power flow calculations. The simulation model is completed with several services that provide the agent model with information about its environment. These services include a *WeatherService*, a *MarketService*, an *EvService* and a *SchedulerService*. While the first three services provide internal and external primary and secondary data from different data sources, the scheduler service is responsible for the temporal synchronization of the occurring events in the agent model.

This thesis designs the model of SIMONA and also implements it as executable simulation. Therefore, different implementation architectures available for agent-based simulation models, in particular Object-oriented Programming (OOP) and actor-based programming, are discussed and evaluated as well. The conceptual similarity between agents and actors makes actor-based programming a suitable and well-fitting implementation approach. Hence, the developed model is implemented as an actor-based simulation that allows for a scalable, asynchronous, and concurrent application on computational clusters or cloud infrastructure.

While the overall simulation model is designed with a research partner, this thesis contributes to some of SIMONA's essential core functions, exclusively reflecting the author's work. These contributions are summarized in the following.

The first contribution is the development of a discrete-event time synchronization and model execution mechanism. The contribution includes a formal description of the responsible scheduling entity, the *SimScheduler* (*SimS*), and its interaction protocol which ensures time-synchronous coordination of all agent entities. Combining agent-based methods with a time representation and synchronization using a discrete-event mechanism results in a flexible and computationally efficient overall model execution. Within the developed approach, a model calculation is only performed when a particular event changes the model state, either locally or globally. This on-demand calculation for state changes of agent or service entities and the omission of points in time without state changes reduces the computational effort. Additionally, as emergent behavior becomes visible only when simulating a large number of entities, there are a variety of use cases where different spatial and temporal scales have to be combined in the same simulation model. This combination is achieved with the designed discrete-event time advancement mechanism. It allows seamless integration and execution of models with different temporal resolutions without further data preparation. Furthermore, the properties of decomposition and decoupling, inherent in the developed approach, allow for a concurrent, efficient entity execution. The mentioned aspects make the developed model scalable, flexible and support its applicability in practice.

The second contribution does not only relate to SIMONA but contributes to power flow calculation research in general. The execution of power flow calculations is crucial in order to calculate node voltages and grid utilization for time series generation. However, large-scale grid simulations and an agent-based setting make them challenging and computationally expensive. Therefore, a new power flow execution approach, a hybrid Distributed Backward-Forward Sweep Algorithm (DBFS) power flow, is developed and integrated into SIMONA's agent model. This approach combines the general idea of a backward-forward sweep methodology with a pre-defined grid partitioning approach, a general Newton-Raphson Method (NR-Method) and an agent-based discrete-event framework. This way, the approach supports the decomposition of power flow equations and their reformulation as closed subproblems, making them subsequently distributable and solvable in an agent-based context. The computability of the power flow benefits twofold from the developed approach. Firstly, the decomposition of the power flow equations into subproblems results in an easier invertibility of the Jacobian, which correspondingly simplifies and accelerates its calculation. Secondly, the applied partitioning allows a distribution over several agent instances. If they are implemented following an actor- or thread-based paradigm, individual grid partitions can be computed asynchronously and concurrently, which can lead to shorter computation times, particularly for large grid structures. The contribution comprises a mathematical formulation of the grid partioning process as well as a formal description of the corresponding agent entity, the *GA*, and its integration into the agent model, including its interaction protocol, which is required to ensure a stable, distributed execution of the developed DBFS. Although designed for an agent-based setting, the developed approach is also usable as a standalone algorithm.

The third contribution is the development, verification and validation of one specific *SPA*, the *PvAgent*. As *SPAs* are generic in SIMONA, the focus is primarily on the description of the physical PV model, which defines the *SPA*'s shape. The presented model combines direct, diffuse and reflected irradiance to allow for a detailed, realistic calculation of the resulting power feed-in. Following an extensive formal description of the developed model, its functionality is verified and validated. The verification is carried out by comparing different calculation results of the previously existing model with the newly developed one. The results indicate that the newly developed model calculates reasonable power feed-in values. In order to validate the model's functionality, an extensive comparison validation is carried out. In this context, the same weather data is used to calculate power feed-in resulting from calculations of the developed model and an already established model. Subsequently, the result data is compared against each other. This comparison shows that the developed model produces valid feed-in results and their quality is sufficient to be applied in SIMONA.

The capabilities of SIMONA are demonstrated in the context of two use cases. The first use case contributes to the validation of the developed model. Several benchmark grids are simulated at their nominal operation point and the result data are compared with available reference results. Although minor deviations occur for some cases, the majority of results show that SIMONA's calculation results are precise enough to be used in the context of power distribution system analysis for planning and operation purposes. In the second use case, a full year time series simulation is executed and analyzed. It demonstrates the consideration of flexibility using the example of a $Q(V)$ voltage setpoint control. There, an initial and two future scenarios of a PV park are analyzed. The analysis revealed that the consideration of flexibility, in this case a $Q(V)$ voltage setpoint control, can positively impact the node voltages and may allow shifting required grid extensions due to a connection request up to the following maintenance window. Furthermore, a comparison between conventional power flow calculations in the context of distribution grid planning is compared to time-series-based analyses. The analysis shows that conventional grid planning tends to overestimate the impact of DER which may result in oversizing of distribution grids. In the context of the use cases, the SIMONA-Ecosystem consisting of several developed OpenSource (OS) tools to support pre- and post-process steps in the application of SIMONA for grid analysis, is also applied and briefly introduced.

Critical Discussion and Outlook

The two use cases validate and demonstrate the functionality of the developed model. However, as there is always room for improvement, the following section critically reflects the overall model and provides an outlook on additional research needs.

First of all, as a simulation model is developed as the main contribution of this thesis, it needs to be stated that models only represent parts of their real-world counterparts. Hence, they inevitably contain model errors by design, which must be kept in mind whenever model runs are executed and their results are analyzed. One way to quantify and remove these errors is to validate the developed model environment for different use cases and scenarios using real-world data or data from other, already validated models. Whenever the existing model is applied or extended in future research, it should also carefully be further validated and improved.

With regard to the discrete-event time representation, the fact that there is only a single *SimS* may become a bottleneck from a performance and scalability perspective for extensive simulations. Especially if additional interaction and communication protocols are implemented, which subsequently lead to an increased number of events that need to be processed, having only a single entity for their handling may not be sufficient anymore. Therefore, further research on the application of distributed scheduling concepts is required. A possible solution would be to have multiple schedulers, e.g., one scheduler per *GA* and all connected *SPAs* in the corresponding partition. Furthermore, concurrent event scheduling may also be a potentially helpful extension of the existing scheduling approach. The current model also does not fully support optimistic scheduling due to the lack of a rollback strategy. Future research may investigate the potential of an entire optimistic scheduling approach and if the performance benefit justifies the increased complexity. However, particularly the strong time interdependencies between agents depending on their use case make realizing this approach quite challenging.

The developed DBFS also has room for improvements. In the current model, only single reference or slack nodes are considered. While this is sufficient for most of the considered distribution grid application cases, having multiple reference nodes may be required in future application cases and the existing model should be extended accordingly. Another aspect that might be worth to be investigated is the application of the algorithm on fully meshed grid structures. Again, while this is not an application case for most distribution grids, meshed structures may particularly occur on higher voltage levels. While the algorithm is generally capable of handling cyclic grid structures, it has not been tested with meshed ones yet and should be tested and validated against them accordingly. Concerning the power flow execution, one may consider investigating if a replacement of the existing NR-Method may improve performance or result accuracy. The flexible implementation of the power flow module allows a simple exchange with alternative approaches, e.g., a DC power flow algorithm. Furthermore, it is also possible to use different implementations, e.g., PyPower or pandapower.

This thesis develops, verifies and validates a new PV model against an existing established one. However, lacking field measurements of weather data and the corresponding PV power feed-in, it was not possible to validate the developed physical PV against real field data. Tuning and further validating the developed model is an important task that should be done as soon as the required data is available. Besides the concrete *PvAgent*, the generic *SPA* also may be extended with additional or improved physical models. Thanks to its newly developed structure, these models can be easily integrated.

Additionally, one may consider evaluating and implementing additional behaviors or control algorithms, e.g., demand-side management approaches or different optimization algorithms. Such approaches or algorithms may be integrated into the existing agent entities or introduced by other, newly developed ones. In any case, the developed interaction protocols allow for an easy extension and integration of such approaches. Furthermore, introducing new model entities to simulate new concepts like regional cellular approaches may be an exciting extension and use case.

Regarding the validation and verification of SIMONA the first presented application case already provides extensive results which demonstrate the model's validity. However, even though most simulated grid models are very close to the reference values, still a few outliers can be observed in the results. Identifying the causes of the outliers and eliminating them could be a good starting point for further improvements of the results of SIMONA. Similar validations could also be carried out with additional grids. The remaining SimBench benchmark grid models, which have not yet been taken into account, offer a good initial starting point.

Executing simulations with similar tools like pandapower, DIgSILENT PowerFactory or NePlan and a subsequent result comparison is also another possibility to further improve and validate SIMONA's model quality. Besides this, another valuable contribution to the research community might be to provide SimBench reference data not only for a single operating point but for a whole year using SIMONA. Such a contribution could provide exemplary grid time series that can be used to validate other time-series simulations and may serve several other purposes, e.g., the development of key indicators for time series-based grid planning or additional post-processing tools.

There is also room for further development regarding the analysis of time series and their utilization for analysis, operation, and planning purposes. The detailed bottom-up approach of the developed simulation model, including individual behaviors, results in a large number of different result data, and this number even increases if several simulations for multiple scenarios are executed. Capturing and evaluating this large amount of data with existing methods is challenging and sometimes impossible. Therefore, investigating new approaches to efficiently work with the simulation results and identifying time series-based key indicators might be another exciting research direction.

Given the increasing importance of OS also in the field of electrical power system simulation, ongoing research efforts currently work on making SIMONA publicly available. Its publication will make it available for applications to the research community and practitioners and represents an important step for further development. Its free availability may support the digitization of DSOs and their distribution system analysis, planning and operation processes.

Closing Remarks

The developed holistic simulation model, SIMONA, is one possible answer to the initially formulated research question. Answering the research question sets the path for applying the developed model for power system investigation purposes. Now following steps should concentrate on model application and its subsequent improvement. Currently, ongoing research projects, like *TRANSENSE*[17] or *MoMeEnT*[18] in which SIMONA is applied can make an essential contribution to this.

[17]Funded by the BMWi (grant number 03EI6044A); SIMONA is used as a simulation-as-a-service for training data generation for neural state estimation.

[18]Funded by the DFG (grant number RE 2930/25-1); SIMONA is applied for socio-technical multi-energy system simulation and analysis.

Developing models to investigate real systems is always an ongoing, iterative process. There are several open paths and questions available for further investigations and the next iteration in the modeling and simulation cycle is almost around the corner. Therefore, the contributions of this thesis are merely a few steps in the context of agent-based power distribution system simulation research. The future will tell whether and how the model developed in this thesis will evolve.

Bibliography

[1] *Erneuerbare-Energien-Gesetz vom 21. Juli 2014 (BGBl. I S. 1066), das zuletzt durch Artikel 1 des Gesetzes vom 21. Dezember 2020 (BGBl. I S. 3138) geändert worden ist.*

[2] *Paris agreement*, 2015. [Online]. Available: https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=_en (visited on 09/29/2021).

[3] T. Strasser, F. Pröstl Andrén, G. Lauss, *et al.*, "Towards holistic power distribution system validation and testing—an overview and discussion of different possibilities," *e & i Elektrotechnik und Informationstechnik*, vol. 134, 2017. DOI: 10.1007/s00502-016-0453-3.

[4] German Association of Energy and Water Industries (BDEW), "Smart Grid Traffic Light Concept," German Association of Energy and Water Industries (BDEW), Berlin, Tech. Rep. March, 2015.

[5] ——, "Konkretisierung des Ampelkonzepts im Verteilungsnetz," German Association of Energy and Water Industries (BDEW), Berlin, Tech. Rep. February, 2017.

[6] *Energiewirtschaftsgesetz vom 7. Juli 2005 (BGBl. I S. 1970, 3621), das zuletzt durch Artikel 3 des Gesetzes vom 18. Mai 2021 (BGBl. I S. 1122) geändert worden ist.*

[7] Y. Bar-Yam, *Dynamics Of Complex Systems.* Westview Press, 1997.

[8] F. Klügl, "Agent-based simulation engineering," Habilitation, University of Würzburg, 2010.

[9] J. Kays, *Agent-based simulation environment for improving the planning of distribution grids.* Göttingen: Sierke, 2014, ISBN: 9-783868-446623.

[10] A. Seack, *Time-series based distribution grid planning considering interaction of network participants with a multi-agent system.* Göttingen: Sierke Verlag, 2016, ISBN: 9-783868-447965.

[11] C. Kittl, *Entwurf und Validierung eines individualitätszentrierten, interdisziplinären Energiesystemsimulators basierend auf ereignisdiskreter Simulation und Agententheorie.* Düren: Shaker Verlag, 2022, ISBN: 978-3-8440-8463-4. DOI: 10.17877/DE290R-22548.

[12] S. Meinecke, D. Sarajlić, S. R. Drauz, *et al.*, “SimBench—a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis,” *Energies*, vol. 13, no. 12, p. 3290, Jun. 2020. DOI: 10.3390/en13123290.

[13] S. Kippelt, *Dezentrale Flexibilitätsoptionen und ihr Beitrag zum Ausgleich der fluktuierenden Stromerzeugung Erneuerbarer Energien*, DE, 1st ed., ser. Dortmunder Beiträge zu Energiesystemen, Energieeffizienz und Energiewirtschaft. Aachen: Shaker Verlag, 2018, vol. 3, ISBN: 978-3-8440-5853-6.

[14] C. Wagner, *Integration und Bewertung der Spitzenkappung als Planungsgrundsatz zur wirtschaftlichen Netzentwicklung in Mittelspannungsnetzen*, 1st ed., ser. Dortmunder Beiträge zu Energiesystemen, Energieeffizienz und Energiewirtschaft. Dortmund: Shaker Verlag, 2019, vol. 6, ISBN: 978-3-8440-6400-1.

[15] G. Celli, F. Pilo, G. G. Soma, *et al.*, “A comparison of distribution network planning solutions: Traditional reinforcement versus integration of distributed energy storage,” in *2013 IEEE Grenoble Conference*, 2013, pp. 1–6. DOI: 10.1109/PTC.2013.6652338.

[16] S. Kippelt, C. Wagner, and C. Rehtanz, “Consideration of new electricity applications in distribution grid expansion planning and the role of flexibility,” in *International ETG Congress 2017*, 2017.

[17] V. Neimane, “On development planning of electricity distribution networks,” PhD Thesis, Royal Institute of Technology, Stockholm, 2001.

[18] A. Keane, L. F. Ochoa, C. L. T. Borges, *et al.*, “State-of-the-art techniques and challenges ahead for distributed generation planning and optimization,” *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1493–1502, 2013. DOI: 10.1109/TPWRS.2012.2214406.

[19] I. Ramirez-Rosado, “Review of distribution system planning models: A model for optimal multistage planning,” English, *IEE Proceedings C (Generation, Transmission and Distribution)*, vol. 133, 397–408(11), 7 Nov. 1986, ISSN: 0143-7046. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-c.1986.0060 (visited on 09/29/2021).

[20] S. Karagiannopoulos, P. Aristidou, A. Ulbig, *et al.*, “Optimal planning of distribution grids considering active power curtailment and reactive power control,” in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, 2016, pp. 1–5. DOI: 10.1109/PESGM.2016.7741538.

[21] Deutsche Energie-Agentur GmbH (dena), “Ausbau- und Innovationsbedarf der Stromverteilnetze in Deutschland bis 2030.,” Deutsche Energie-Agentur GmbH (dena), Berlin, Tech. Rep., 2012, p. 410. [Online]. Available: https://www.dena.de/fileadmin/dena/Dokumente/Pdf/9100_dena-Verteilnetzstudie_Abschlussbericht.pdf (visited on 09/29/2021).

[22] J. Büchner, J. Katzfey, O. Flörcken, *et al.*, “Moderne Verteilernetze für Deutschland,” Tech. Rep., 2014. [Online]. Available: https://www.bmwi.de/Redaktion/DE/Publikationen/Studien/verteilernetzstudie.pdf (visited on 09/29/2021).

[23] E. Tønne, J. A. Foosnes, and T. Pynten, “Power system planning in distribution networks today and in the future with smart grids,” in *Proceedings of the 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, 2013. DOI: 10.1049/cp.2013.1233.

[24] G. Roupioz, X. Robe, and F. Gorgette, “First use of smart grid data in distribution network planning,” in *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, 2013. DOI: 10.1049/cp.2013.0835.

[25] *Netzausbaubeschleunigungsgesetz Übertragungsnetz vom 28. Juli 2011 (BGBl. I S. 1690), das zuletzt durch Artikel 4 des Gesetzes vom 25. Februar 2021 (BGBl. I S. 298) geändert worden ist.*

[26] R. Broll, C. Jahns, F. Kurtz, *et al.*, “Digitale Systeme und Dienste für die Energiesystemtransformation,” Dortmund, Tech. Rep. DOI: 10.17877/DE290R-21909.

[27] D. Giavarra, “Application of time-resolved input data for smart grid simulation,” English, *CIRED - Open Access Proceedings Journal*, vol. 2017, 2106–2109(3), 1 Oct. 2017. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/oap-cired.2017.0380 (visited on 09/29/2021).

[28] H. Çakmak, A. Erdmann, M. Kyesswa, *et al.*, “A new distributed co-simulation architecture for multi-physics based energy systems integration,” *Automatisierungstechnik*, vol. 67, no. 11, pp. 972–983, 2019, 37.06.01; LK 01, ISSN: 2196-677X, 0178-2312. DOI: 10.1515/auto-2019-0081.

[29] G. Tian, Y. Gu, D. Shi, *et al.*, "Neural-network-based power system state estimation with extended observability," *Journal of Modern Power Systems and Clean Energy*, pp. 1–11, 2021. DOI: 10.35833/MPCE.2020.000362.

[30] V. Crastan and D. Westermann, *Elektrische Energieversorgung 3*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-20100-4.

[31] E. Handschin, *Elektrische Energieübertragungssysteme*, vol. 2, p. 295, ISBN: 3-7785-1401-6.

[32] K. Heuck, K.-D. Dettmann, and D. Schulz, *Elektrische Energieversorgung - Erzeugung, Übertragung und Verteilung elektrischer Energie für Studium und Praxis*, 9. Edition. Berlin Heidelberg New York: Springer-Verlag, 2013, ISBN: 978-3834816993.

[33] A. J. Schwab, *Elektroenergiesysteme - Erzeugung, Transport, Übertragung und Verteilung elektrischer Energie*, 6. Edition. Berlin Heidelberg New York: Springer-Verlag, 2019, p. 858, ISBN: 978-3662603734.

[34] J. Das, *Power System Analysis: Short-circuit Load Flow and Harmonics*, Second, ser. Power Engineering. CRC Press, 2012, ISBN: 9781351825146.

[35] W. F. Tinney and C. E. Hart, "Power flow solution by newton's method," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-86, no. 11, pp. 1449–1460, 1967. DOI: 10.1109/TPAS.1967.291823.

[36] B. Stott and O. Alsac, "Fast decoupled load flow," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-93, no. 3, pp. 859–869, 1974. DOI: 10.1109/TPAS.1974.293985.

[37] J. Rupa and S. Ganesh, "Power flow analysis for radial distribution system using backward/forward sweep method," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 1621–1625, 2014.

[38] W. Kersting, *Distribution System Modeling and Analysis, Third Edition*. Taylor & Francis, 2012, ISBN: 9781439856222.

[39] D. Thukaram, H. Wijekoon Banda, and J. Jerome, "A robust three phase power flow algorithm for radial distribution systems," *Electric Power Systems Research*, vol. 50, no. 3, pp. 227–236, 1999, ISSN: 0378-7796. DOI: 10.1016/S0378-7796(98)00150-3.

[40] A. Trias, "The holomorphic embedding load flow method," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8. DOI: 10.1109/PESGM.2012.6344759.

[41] M. Fikri, B. Cheddadi, O. Sabri, *et al.*, "Power flow analysis by numerical techniques and artificial neural networks," in *2018 Renewable Energies, Power Systems Green Inclusive Economy (REPS-GIE)*, 2018, pp. 1–5. DOI: 10.1109/REPSGIE.2018.8488870.

[42] R. Idema, D. Lahaye, and C. Vuik, "Load flow literature survey," Tech. Rep., 2009. [Online]. Available: http://ta.twi.tudelft.nl/TWA_Reports/09/09-04.pdf (visited on 09/29/2021).

[43] L. Braz, C. Castro, and C. Murati, "A critical evaluation of step size optimization based load flow methods," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 202–207, 2000. DOI: 10.1109/59.852122.

[44] M. Baran and F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on Power Delivery*, vol. 4, no. 1, pp. 725–734, 1989. DOI: 10.1109/61.19265.

[45] M. Baran and F. Wu, "Optimal sizing of capacitors placed on a radial distribution system," *IEEE Transactions on Power Delivery*, vol. 4, no. 1, pp. 735–743, 1989. DOI: 10.1109/61.19266.

[46] D. K. Molzahn and I. A. Hiskens, *A Survey of Relaxations and Approximations of the Power Flow Equations*, 1-2. 2019, vol. 4. DOI: 10.1561/3100000012.

[47] FGH GmbH. "Integral." (Sep. 20, 2021), [Online]. Available: https://www.fgh-ma.de/de/portfolio-produkte/software/netzberechnung-mit-integral (visited on 09/29/2021).

[48] NePlan AG. "NePlan - Power System Analysis." (Sep. 29, 2021), [Online]. Available: https://www.neplan.ch/ (visited on 09/29/2021).

[49] DIgSILENT GmbH. "PowerFactory." version 2021 Service Pack 1. (2021), [Online]. Available: https://www.digsilent.de/de/powerfactory.html (visited on 09/29/2021).

[50] PowerWorld Corporation. "PowerWorld Simulator." (Jun. 26, 2021), [Online]. Available: https://www.powerworld.com/ (visited on 09/29/2021).

[51] Siemens AG. "PSS®SINCAL – Software zur Analyse und Planung aller Arten von Energieversorgungsnetzen." (Sep. 29, 2021), [Online]. Available: https://new.siemens.com/de/de/produkte/energie/energieautomatisierung-und-smart-grid/pss-software/pss-sincal.html (visited on 06/26/2021).

[52] Adaptricity AG. “Adaptricity.Plan.” (Sep. 29, 2021), [Online]. Available: https://www.adaptricity.com/adaptricity-plan/ (visited on 09/12/2021).

[53] envelio GmbH. “Intelligent Grid Platform.” (Sep. 12, 2021), [Online]. Available: https://envelio.com/de/igp/ (visited on 09/29/2021).

[54] R. D. Zimmerman and C. E. Murillo-Sánchez, *MATPOWER*, version 7.1, Oct. 2020. DOI: 10.5281/zenodo.4074135.

[55] F. Milano, “An open source power system analysis toolbox,” *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1199–1206, 2005. DOI: 10.1109/TPWRS.2005.851911.

[56] The MathWorks, Inc. “Matlab.” (Sep. 29, 2021), [Online]. Available: https://www.mathworks.com/products/matlab.html (visited on 09/29/2021).

[57] Free Software Foundation. “GNU Octave.” (Sep. 29, 2021), [Online]. Available: www.gnu.org/software/octave/ (visited on 09/29/2021).

[58] R. Lincoln. “PYPOWER-Repository on GitHub.” (Sep. 29, 2021), [Online]. Available: https://github.com/rwl/PYPOWER (visited on 09/29/2021).

[59] T. Brown, J. Hörsch, and D. Schlachtberger, “Pypsa: Python for power system analysis,” *Journal of Open Research Software*, vol. 6, no. 4, 1 2018. DOI: 10.5334/jors.188. eprint: 1707.09913.

[60] L. Thurner, A. Scheidler, F. Schäfer, *et al.*, “Pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems,” *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Nov. 2018, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2018.2829021.

[61] S. P. Vera. “GridCal-Repository on GitHub.” (Sep. 29, 2021), [Online]. Available: https://github.com/SanPen/GridCal (visited on 09/29/2021).

[62] S. Pfenninger, L. Hirth, I. Schlecht, *et al.*, “Opening the black box of energy modelling: Strategies and lessons learned,” *Energy Strategy Reviews*, vol. 19, pp. 63–71, 2018, ISSN: 2211-467X. DOI: 10.1016/j.esr.2017.12.002.

[63] D. Lohmeier, D. Cronbach, S. R. Drauz, *et al.*, “Pandapipes: An open-source piping grid calculation package for multi-energy grid simulations,” *Sustainability*, vol. 12, no. 23, 2020, ISSN: 2071-1050. DOI: 10.3390/su12239899.

[64] D. Montenegro and R. C. Dugan, “OpenDSS and OpenDSS-PM open source libraries for NI LabVIEW,” in *2017 IEEE Workshop on Power Electronics and Power Quality Applications (PEPQA)*, IEEE, May 2017. DOI: 10.1109/pepqa.2017.7981639.

[65] D. Montenegro, M. Hernandez, and G. A. Ramos, “Real time OpenDSS framework for distribution systems simulation and analysis,” in *2012 Sixth IEEE/PES Transmission and Distribution: Latin America Conference and Exposition (T&D-LA)*, IEEE, Sep. 2012. DOI: 10.1109/tdc-la.2012.6319069.

[66] A. Latif, M. Ikechi, I. Hawaiian Electric Company, *et al.*, *NREL/PyDSS*, Dec. 2018. DOI: 10.11578/dc.20190514.1.

[67] D. P. Chassin, J. C. Fuller, and N. Djilali, “GridLAB-D: An agent-based simulation framework for smart grids,” May 13, 2014. arXiv: 1405.3136.

[68] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, “GridLAB-D: An open-source power systems modeling and simulation environment,” in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, IEEE, Apr. 2008. DOI: 10.1109/tdc.2008.4517260.

[69] H. Meier, C. Fünfgeld, T. Adam, *et al.*, “Repräsentative VDEW-Lastprofile,” Brandenburgisch Technische Universität Cottbus, Cottbus, Tech. Rep., 1999. [Online]. Available: https://www.bdew.de/media/documents/1999_Repraesentative-VDEW-Lastprofile.pdf (visited on 09/29/2021).

[70] S. Pfenninger and I. Staffell, “Long-term patterns of european PV output using 30 years of validated hourly reanalysis and satellite data,” *Energy*, vol. 114, pp. 1251–1265, Nov. 2016. DOI: 10.1016/j.energy.2016.08.060.

[71] ——, “Renewables.Ninja.” (Sep. 29, 2021), [Online]. Available: https://www.renewables.ninja/ (visited on 09/29/2021).

[72] M. Müller, F. Biedenbach, and J. Reinhard, “Development of an integrated simulation model for load and mobility profiles of private households,” *Energies*, vol. 13, no. 15, 2020, ISSN: 1996-1073. DOI: 10.3390/en13153843.

[73] C. Wagner, C. Waniek, and U. Häger, “Modeling of household electricity load profiles for distribution grid planning and operation,” in *2016 IEEE International Conference on Power System Technology (POWERCON)*, 2016, pp. 1–6. DOI: 10.1109/POWERCON.2016.7753889.

[74] N. Pflugradt and B. Platzer, "Behavior based load profile generator for domestic hot water and electricity use," in *12th International Conference on Energy Storage (Innostock), Lleida, Spain*, 2012.

[75] S. Meinecke, L. Thurner, and M. Braun, "Review of steady-state electric power distribution system datasets," *Energies*, vol. 13, no. 18, p. 4826, Sep. 2020. DOI: 10.3390/en13184826.

[76] S. Ciraci, J. Daily, J. Fuller, *et al.*, "Fncs: A framework for power system and communication networks co-simulation," in *Proceedings of the Symposium on Theory of Modeling and Simulation - DEVS Integrative*, ser. DEVS '14, Tampa, Florida: Society for Computer Simulation International, 2014.

[77] C. Steinbrink, S. Lehnhoff, S. Rohjans, *et al.*, "Simulation-based validation of smart grids – status quo and future research trends," in *Industrial Applications of Holonic and Multi-Agent Systems*, Cham: Springer International Publishing, 2017, pp. 171–185, ISBN: 978-3-319-64635-0.

[78] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids," in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, 2011, pp. 55–60. DOI: 10.1109/SGMS.2011.6089027.

[79] B. Palmintier, D. Krishnamurthy, P. Top, *et al.*, "Design of the helics high-performance transmission-distribution-communication-market co-simulation framework," in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2017, pp. 1–6. DOI: 10.1109/MSCPES.2017.8064542.

[80] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, *et al.*, "Cpes testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, 2019, ISSN: 2076-3417. DOI: 10.3390/app9050923.

[81] A. Meintz, T. Lipman, B. Palmintier, *et al.*, "Grid-Enhanced, Mobility-Integrated Network Infrastructures for Extreme Fast Charging (GEMINI-XFC)," Jun. 2021. [Online]. Available: https://www.energy.gov/sites/default/files/2021-07/elt258_meintz_2021_o_5-28_225pm_LR_ML.pdf (visited on 09/29/2021).

[82] A. Maria, "Introduction to modeling and simulation," in *Proceedings of the 29th Conference on Winter Simulation*, ser. WSC '97, Atlanta, Georgia, USA: IEEE Computer Society, 1997, pp. 7–13, ISBN: 078034278X. DOI: 10.1145/268437.268440.

[83] G. Schweiger, H. Nilsson, J. Schoeggl, *et al.*, *Modeling and simulation of large-scale systems: A systematic comparison of modeling paradigms*, 2019. arXiv: 1909.00484.

[84] J. Banks, J. S. Carson, B. L. Nelson, *et al.*, *Discrete-Event System Simulation*, 5th. Pearson, Jul. 2010, ISBN: 9780136062127.

[85] B. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation - Discrete Event & Iterative System Computational Foundations*, 3rd. Amsterdam, Boston: Academic Press, 2018, ISBN: 978-0-128-13407-8.

[86] A. M. Law, *Simulation Modeling and Analysis*, 5th. New York: McGraw-Hill Education, 2015, ISBN: 978-0-073-40132-4.

[87] J. W. Schmidt and T. R. Edward, *Simulation and analysis of industrial systems*. R. D. Irwin Homewood, Ill, 1970.

[88] P. A. Fishwick, *Simulation Model Design and Execution: Building Digital Worlds*, 1st. USA: Prentice Hall PTR, 1995, ISBN: 0130986097.

[89] Y. Monsef and E. Kerckhoffs, *Modelling and Simulation of Complex Systems: Concepts, Methods and Tools*, ser. Frontiers in Simulation. Society for Computer Simulation, 1997, ISBN: 9781565551183.

[90] J. A. Sokolowski and C. M. Banks, *Modeling and Simulation Fundamentals*. New York: John Wiley & Sons, Inc., 2010, ISBN: 978-0-470-59061-4. DOI: 10.1007/978-3-030-18869-6_2.

[91] J. Nyboer, "Simulating evolution of technology: An aid to energy policy analysis," PhD Thesis, 1997.

[92] F. Sensfuß, "Assessment of the impact of renewable electricity generation on the german electricity sector: An agent-based simulation approach," PhD Thesis, 2007. DOI: 10.5445/IR/1000007777.

[93] M. Ventosa, A. Baillo, A. Ramos, *et al.*, "Electricity market modeling trends," *Energy Policy*, vol. 33, no. 7, pp. 897–913, 2005.

[94] A. Borshchev and A. Filippov, "From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools," in *22nd International Conference of the System Dynamics Society*, 2004.

[95] A. Sulistio, C. S. Yeo, and R. Buyya, "A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools," *Software - Practice and Experience*, vol. 34, no. 7, pp. 653–673, 2004, ISSN: 00380644. DOI: 10.1002/spe.585.

[96] K. G. Troitzsch, *Modellbildung und Simulation in den Sozialwissenschaften*. VS Verlag für Sozialwissenschaften, 1990. DOI: 10.1007/978-3-322-93561-8.

[97] S. Robinson, *Simulation - The Practice of Model Development and Use*. New York: Macmillan Education UK, 2014, ISBN: 978-1-137-32802-1.

[98] M. Pidd, *Computer Simulation in Management Science*, English, 5th. John Wiley and Sons Ltd, 2006, ISBN: 978-0-470-09230-9.

[99] E. J. Derrick, O. Balci, and R. E. Nance, “A comparison of selected conceptual frameworks for simulation modeling,” in *Proceedings of the 21st Conference on Winter Simulation*, ser. WSC ’89, Washington, D.C., USA: Association for Computing Machinery, 1989, pp. 711–718, ISBN: 0911801588. DOI: 10.1145/76738.76829.

[100] K. D. Tocher, *The Art of Simulation*. English Universities Press, 1967.

[101] K. S. Perumalla and R. M. Fujimoto, “Efficient large-scale process-oriented parallel simulations,” in *Proceedings of the 30th Conference on Winter Simulation*, ser. WSC ’98, Washington, D.C., USA: IEEE Computer Society Press, 1998, pp. 459–466, ISBN: 0780351347.

[102] A. Kunert, “Optimistic parallel process-oriented des in java using bytecode rewriting,” *Proc. of MESM 2008*, pp. 15–21, 2008.

[103] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943, ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[104] V. R. Lesser and L. D. Erman, “Distributed interpretation: A model and experiment,” in *Distributed Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, pp. 120–139, ISBN: 093461363X.

[105] M. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd. Wiley Publishing, 2009, ISBN: 0470519460.

[106] C. M. Macal, “Everything you need to know about agent-based modelling and simulation,” *Journal of Simulation*, vol. 10, no. 2, pp. 144–156, 2016, ISSN: 17477786. DOI: 10.1057/jos.2016.7.

[107] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1999, ISBN: 0201360489.

[108] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th, ser. Pearson series in artificial intelligence. Pearson, 2020, ISBN: 0134610997.

[109] C. V. Pal, F. Leon, M. Paprzycki, *et al.*, "A Review of Platforms for the Development of Agent Systems," *arXiv*, no. July 2020, pp. 1–40, 2020, ISSN: 23318422. arXiv: 2007.08961.

[110] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995. DOI: 10.1017/S0269888900008122.

[111] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *Intelligent Agents III Agent Theories, Architectures, and Languages*, J. P. Müller, M. J. Wooldridge, and N. R. Jennings, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 21–35, ISBN: 978-3-540-68057-4.

[112] H. S. Nwana, "Software agents: An overview," *The Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996. DOI: 10.1017/S026988890000789X.

[113] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence*, vol. 60, no. 1, pp. 51–92, 1993, ISSN: 0004-3702. DOI: 10.1016/0004-3702(93)90034-9.

[114] D. Kafura and J.-P. Briot, "Actors And Agents," *IEEE Concurrency*, vol. 6, no. 2, pp. 24–29, 1998. DOI: 10.1109/MCC.1998.678786.

[115] Foundation for Intelligent Physical Agents (FIPA). "Fipa standard specifications." (2002), [Online]. Available: http://www.fipa.org/repository/standardspecs.html (visited on 09/29/2021).

[116] C. Rehtanz, *Autonomous Systems and Intelligent Agents in Power System Control and Operation*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-662-05955-5.

[117] U. Häger, "Agent-based real-time coordination of power flow controllers," PhD Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, 2013, ISBN: 978-3-86844-507-7.

[118] S. Lehnhoff, *Dezentrales vernetztes Energiemanagement: Ein Ansatz auf Basis eines verteilten adaptiven Realzeit-Multiagentensystems*, ser. Vieweg+Teubner research. Vieweg+Teubner Verlag, 2010, ISBN: 9783834896582.

[119] M. W. Khan and J. Wang, "The research on multi-agent system for microgrid control and optimization," *Renewable and Sustainable Energy Reviews*, vol. 80, pp. 1399–1411, 2017, ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.05.279.

[120] A. Kantamneni, L. E. Brown, G. Parker, *et al.*, “Survey of multi-agent systems for microgrid control,” *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 192–203, 2015, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2015.07.005.

[121] J. R. Pesente, J. G. Rolim, and M. Moreto, “Multiagent systems in power system protection: Review, classification and perspectives,” *IEEE Latin America Transactions*, vol. 14, no. 7, pp. 3285–3290, 2016. DOI: 10.1109/TLA.2016.7587632.

[122] C. Macal and M. North, “Introductory tutorial: Agent-based modeling and simulation,” in *Proceedings of the Winter Simulation Conference 2014*, 2014, pp. 6–20. DOI: 10.1109/WSC.2014.7019874.

[123] B. Logan, “An agent programming manifesto,” *International Journal of Agent-Orientated Software Engineering*, vol. 6, pp. 187–210, 2 2018, ISSN: 1746-1375.

[124] R. C. Cardoso and A. Ferrando, “A review of agent-based programming for multi-agent systems,” *Computers*, vol. 10, no. 2, 2021, ISSN: 2073-431X. DOI: 10.3390/computers10020016.

[125] P. Wegner, “Concepts and paradigms of object-oriented programming,” *SIGPLAN OOPS Mess.*, vol. 1, no. 1, pp. 7–87, 1990, ISSN: 1055-6400. DOI: 10.1145/382192.383004.

[126] C. Hewitt, P. Bishop, and R. Steiger, “A universal modular actor formalism for artificial intelligence,” in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, ser. IJCAI’73, Stanford, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235–245.

[127] J.-J. C. Meyer, M. D. Ryan, and H.-D. Ehrich, “Objects, Agents, and Features: An Introduction,” in *Objects, Agents, and Features*, M. D. Ryan, J.-J. C. Meyer, and H.-D. Ehrich, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–7, ISBN: 978-3-540-25930-5.

[128] D. G. Kafura, M. Mukherji, and G. R. Lavender, “Act++ 2.0: A class library for concurrent programming in c++ using actors,” USA, Tech. Rep., 1992.

[129] J. Armstrong, *Programming Erlang: Software for a Concurrent World*, ser. Pragmatic Bookshelf Series. Pragmatic Bookshelf, 2007, ISBN: 9781934356005.

[130] J. Hunt, "Introduction to akka actors," in *A Beginner's Guide to Scala, Object Orientation and Functional Programming*. Cham: Springer International Publishing, 2014, pp. 383–398, ISBN: 978-3-319-06776-6. DOI: 10.1007/978-3-319-06776-6_36.

[131] P. Anantharaman, J. P. Brady, P. Flathers, *et al.*, "Going dark: A retrospective on the north american blackout of 2038," in *Proceedings of the New Security Paradigms Workshop*, ser. NSPW '18, Windsor, United Kingdom: Association for Computing Machinery, 2018, pp. 52–63, ISBN: 9781450365970. DOI: 10.1145/3285002.3285011.

[132] R. C. Cardoso, M. R. Zatelli, J. F. Hübner, *et al.*, "Towards benchmarking actor- and agent-based programming languages," in *Proceedings of the 2013 Workshop on Programming Based on Actors, Agents, and Decentralized Control*, ser. AGERE! 2013, Indianapolis, Indiana, USA: Association for Computing Machinery, 2013, pp. 115–126, ISBN: 9781450326025. DOI: 10.1145/2541329.2541339.

[133] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st. O'Reilly Media, Feb. 2015, p. 280, ISBN: 978-1491950357.

[134] J. Rolink, *Modellierung und Systemintegration von Elektrofahrzeugen aus Sicht der elektrischen Energieversorgung*, ser. Reihe ie3 - Institut für Energiesysteme, Energieeffizienz und Energiewirtschaft. Sierke, 2013, ISBN: 9783868445343.

[135] P. A. Lopez, M. Behrisch, L. Bieker-Walz, *et al.*, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[136] A. Horni, K. Nagel, and K. Axhausen, Eds., *Multi-Agent Transport Simulation MATSim*. London: Ubiquity Press, Aug. 2016, p. 618, ISBN: 978-1-909188-75-4, 978-1-909188-76-1, 978-1-909188-77-8, 978-1-909188-78-5. DOI: 10.5334/baw.

[137] R. A. Waraich, "Agent-based simulation of electric vehicles. design and implementation of a framework," en, Ph.D. dissertation, ETH Zurich, Zürich, 2013. DOI: 10.3929/ethz-a-010111112.

[138] G. H. Mealy, "A method for synthesizing sequential circuits," *The Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955. DOI: 10.1002/j.1538-7305.1955.tb03788.x.

[139] M. Kleinberg, K. Miu, and C. Nwankpa, “Distributed Multi-Phase Distribution Power Flow: Modeling, Solution Algorithm and Simulation Results,” *SIMULATION*, vol. 84, no. 8-9, pp. 403–412, 2008. DOI: 10.1177/0037549708098121.

[140] H. Häberlin, *Photovoltaik: Strom aus Sonnenlicht für Verbundnetz und Inselanlagen*. Electrosuisse-Verlag, 2010, ISBN: 380073205X.

[141] J. A. Duffie and W. A. Beckman, *Solar Engineering of Thermal Processe*, 4th. John Wiley & Sons, Inc., Apr. 2013, p. 928. DOI: 10.1002/9781118671603.

[142] R. Perez, R. Seals, P. Ineichen, *et al.*, “A new simplified version of the perez diffuse irradiance model for tilted surfaces,” *Solar Energy*, vol. 39, no. 3, pp. 221–231, 1987, ISSN: 0038092X. DOI: 10.1016/S0038-092X(87)80031-2.

[143] D. R. Myers, *Solar radiation: Practical modeling for renewable energy applications*. 2017, pp. 1–159, ISBN: 9781466503274. DOI: 10.1201/b13898.

[144] S. A. M. Maleki, H. Hizam, and C. Gomes, “Estimation of hourly, daily and monthly global solar radiation on inclined surfaces: Models re-visited,” *Energies*, vol. 10, no. 1, 2017, ISSN: 19961073. DOI: 10.3390/en10010134.

[145] M. M. Rienecker, M. J. Suarez, R. Gelaro, *et al.*, “Merra: Nasa’s modern-era retrospective analysis for research and applications,” *Journal of Climate*, vol. 24, no. 14, pp. 3624–3648, 2011. DOI: 10.1175/JCLI-D-11-00015.1.

[146] D. Sarajlic and C. Rehtanz, “Analysis of the electrical properties of SimBench low voltage benchmark network models,” in *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, IEEE, Oct. 2020. DOI: 10.1109/isgt-europe47291.2020.9248706.

[147] SimBench Research Consortium. “SimBench-documentation.” version EN-1.1.0. (Sep. 7, 2021), [Online]. Available: https://simbench.de/wp-content/uploads/2021/09/simbench_documentation_en_1.1.0.pdf (visited on 09/29/2021).

[148] C. Rehtanz, M. Greve, U. Häger, *et al.*, “Verteilnetzstudie für das land baden"=württemberg,” ef.Ruhr GmbH, Studie, Apr. 13, 2017, 144 pp. [Online]. Available: https://um.baden-wuerttemberg.de/fileadmin/redaktion/m-um/intern/Dateien/Dokumente/5_Energie/Versorgungssicherheit/170413_Verteilnetzstudie_BW.pdf (visited on 09/06/2017).

[149] J. W. Spencer, “Fourier series representation of the position of the sun,” *Search*, no. 5, May 1971.

[150] *DIN 5034-1:2019-12, Tageslicht in Innenräumen - Teil 1: Begriffe und Mindestanforderungen.* DOI: 10.31030/3102732.

[151] Z. Wang, “Chapter 2 - the solar resource and meteorological parameters,” in *Design of Solar Thermal Power Plants*, Z. Wang, Ed., Academic Press, 2019, pp. 47–115, ISBN: 978-0-12-815613-1. DOI: 10.1016/B978-0-12-815613-1.00002-X.

[152] F. Hausdorf, “Über die Absorption des Lichtes in der Atmosphäre (Habilitationsschrift),” in *Felix Hausdorff - Gesammelte Werke Band 5: Astronomie, Optik und Wahrscheinlichkeitstheorie*, J. Bemelmans, E. Brieskorn, C. Binder, *et al.*, Eds., Springer, 2006. DOI: 10.1007/3-540-30669-2.

[153] D. Reinert, F. Prill, H. Frank, *et al.*, “DWD Database Reference for the Global and Regional ICON and ICON-EPS Forecasting System,” Deutscher Wetterdienst (DWD), Tech. Rep. 2.1.4, 2021. [Online]. Available: https://www.dwd.de/DWD/forschung/nwv/fepub/icon_database_main.pdf (visited on 09/29/2021).

[154] S. Meinecke, N. Bornhorst, and M. Braun, “Power system benchmark generation methodology,” in *NEIS 2018; Conference on Sustainable Energy Supply and Energy Storage Systems*, VDE, 2018, pp. 1–6.

[155] C. Spalthoff, D. Sarajlić, C. Kittl, *et al.*, “Simbench: Open source time series of power load, storage and generation for the simulation of electrical distribution grids,” in *Internationaler ETG-Kongress 2019*, May 8, 2019.

List of Abbreviations

ABM	Agent-based Model
APL	Agent-oriented Programming Language
ABS	Agent-based Simulation
MAS	Multi-Agent System
DSO	Distribution Grid Operator
DES	Discrete-Event Modeling and Simulation
DWD	German Meteorological Service
DEVN	Discrete-Event Specified Network
DEVS	Discrete-Event System Specification
EEG	German Renwable Energy Sources Act
DER	distributed energy resources
DBFS	Distributed Backward-Forward Sweep Algorithm
FIPA	Foundation for Intelligent Physical Agents
FSM	Finite-State Machine
GA	*GridAgent*
ICT	Information and Communication Technology
IQR	Interquantile Range
NR-Method	Newton-Raphson Method
OOP	Object-oriented Programming
OS	OpenSource
OC	OpenCore
PV	Photovoltaic
PDS	*Primary Data Service*
PSDM	PowerSystemDatamodel
RMSE	Root Mean Square Error
RN	Renewables.Ninja
SDS	*Secondary Data Service*
SPA	*SystemParticipantAgent*
SoC	State of Charge
SimS	*SimScheduler*
TSO	Transmission Grid Operator
EV	Electric Vehicle
WEC	Wind Energy Converter

List of Symbols

General Notation

$\mathbb{N}$	Set of Natural Numbers
$\mathbb{N}^+$	Set of Positive Natural Numbers
$\mathbb{R}$	Set of Real Numbers
$\mathbb{R}^+_{0,\infty}$	Set of Positive Real Numbers including 0
$\cdot^*$	Conjugate Complex
$\langle x_1, x_2, ..., x_n \rangle$	Tuple of Elements
$[\,\cdot\,]^{m \times n}$	$m \times n$-Matrix
$[J_f\,(\vec{x})]$	Matrix of Partial Derivatives or Jacobian of a function $f(\cdot)$ at $\vec{x}$
$\mathrm{j}\cdot$	Imaginary Quantity Unit
$\underline{\cdot}$	Complex Quantity
$\vec{\cdot}$	Vectorial Quantity

Mathematical Operators

$\Re\{\cdot\}$	Real Part of a Complex Number
$\Im\{\cdot\}$	Imaginary Part of a Complex Number
$\circ$	Hadamard Product (Element-by-Element Matrix Multiplication)
$[\,\cdot\,]^\mathsf{T}$	Transpose of a Matrix
$[\,\cdot\,]^{-1}$	Matrix Inversion
$\lvert\,\cdot\,\rvert$	Element Amount
∂	Partial Derivative of a Function

General Indices

(k)	Iteration Counter

Power Systems

$\mathbf{N}$	Set of Electric Grid Nodes
$n_1, n_2, ..., n_m$	Electric Grid Node Instances
$\mathbf{B}$	Set of Electric Grid Branches
$b_1, b_2, ..., b_l$	Electric Grid Branch Instances
S	Apparent Power
s	Referenced Apparent Power
p	Referenced Active Power

q	Referenced Reactive Power
v	Referenced Voltage Magnitude
i	Referenced Current Magnitude
y	Referenced Admittance
$\underline{y}_{\mathrm{N}}$	Node Admittance Matrix Element
$[\underline{Y}_{\mathrm{N}}]$	Node Admittance Matrix
δ	Voltage Angle
ε	Convergence Barrier
$\vec{e}$	Real Part of the Complex Voltage
$\vec{f}$	Imaginary Part of the Complex Voltage
E	Solar Radiation
$\dot{Q}$	Thermal Power
ϑ	Temperature
$\cos(\varphi)$	Nominal Power Factor
ρ	Surroundings Albedo
η	Inverter Efficiency Factor
α_e	Azimuth Angle
γ_e	Photovoltaik Module Elevation Angle
ω	Hour Angle
AM	Airmass
I_0	Extraterrestrial Radiation
Δ	Brightness Index
θ_z	Zenith Angle
ϵ	Clearness Parameter
J	Day Angle
δ	Sun Declination Angle
α_{s}	Solar Altitude Angle
θ_g	Angle of Incidence
ϕ	Latitude
λ	Longitude
R_{E}	Earth Radius

Power System Indices

G	Power Generation
L	Power Consumption (Load)
g	Electric Grid Instance
i, j	Electric Grid Node Instance
ij	Element between Node i and j
$x0$	Element-x-to-Ground Potential

beam	Beam Radiation
diff	Diffus
e	Irradiance Identifier
h	Horizontal
loss	Element Losses
r	Rated
SR	Sunrise
SS	Sunset

Agent-based Modeling

ABM	Agent-based Model
T	ABM Virtual Time Representation
$ENVM$	ABM Environment Model
AM	ABM Multi-Agent System Model
RM	$ENVM$ Resource Model
GM	$ENVM$ Global Properties
act_{env}	$ENVM$ Environmental Actions Select Function
ex	$ENVM$ Execution Function

Discrete-Event Systems

M	Discrete-Event Model (DES)
$\mathbf{X}$	Set of Input Events
$\mathbf{Y}$	Set of Output Events
$\mathbf{S}$	Set of States
$\mathbf{A}$	Set of Agent States
$\mathbf{A}$	Set of Agent Model States
$\mathbf{Q}$	Set of Total States of a DES Model
δ_{ext}	DES Model External State Transition Function
δ_{int}	DES Model Internal State Transition Function
λ	DES Model Output Function
ta	DES Model Time Advancement Function
s	Single DES Model State
x	Single DES Input Event
y	Single DES Output Event
e	Elapsed Time in a DES Model
t	Single Tick (Logical Timestep)
N	Network of DES
$\mathbf{D}$	Set of DES Component References

$\mathbf{I}$	DES Influencer Set in a DES Network
Select	DES Network tie-breaking Select Function
Z	DES Network Event Mapping Function

Discrete-Event Systems Indices

d	Single DES Component Reference
i	Component Input Values Interface

SIMONA Specific

$\mathbf{T}$	Set of Logical Timepoints
$t_1, t_2, ..., t_n$	Single Logical Timepoints
$\mathbf{D}$	Set of Primary Data
$d_1, d_2, ..., d_n$	Single Primary Data Points
$\mathbf{G}$	Set of Geographical Coordinates
$g_1, g_2, ..., g_n$	Single Geographic Coordinates
$\mathbf{W}$	Set of Weather Data
$w_1, w_2, ..., w_n$	Single Weather Data Points
$\mathbf{P}$	Set of Market Prices
$p_1, p_2, ..., p_n$	Single Market Prices
$\mathbf{E}$	Set of Electric Vehicle Data Points
$e_1, e_2, ..., e_n$	Single Electric Vehicle Data Points
σ_{SoC}	Storage Level
$\sigma_{\mathrm{SoC,th}}$	Thermal Storage Level
$\vec{e}(\cdot)$	SIMONA Environment
$\vec{\rho}(\cdot)$	Primary Input Data
$\vec{\sigma}(\cdot)$	Secondary Input Data
$\vec{\lambda}(\cdot)$	General Agent Communication Request
$\vec{y}(\cdot)$	Agent Output
$\vec{f}(\cdot)$	Agent Next Action Decision Function
$\vec{g}(\cdot)$	Physical Model Transformation Function
$\vec{u}(\cdot)$	Agent's current Utility
$\vec{s}_{\mathrm{m}}(\cdot)$	Physical Model State
S_{m}	Physical Model Store
$U(\cdot)$	Agent Utility Function
$S\mathrm{u}$	Agent Utility Function Value Store
$\mathbf{PSD}$	Set of Primary Data Services
$\mathbf{SDS}$	Set of Secondary Data Services
$\mathbf{A}$	Set of Agent References

$\mathbf{L}$	Sorted List or Queue of Scheduler Events
$\xi(\cdot)$	Scheduler Event Time to Output Mapping Function
ε_S	Simulation Scheduler Optimistic Scheduler Value
ε_{sweep}	DBFS-Sweep Threshold
i_{sweep}	DBFS-Sweep Counter
t_{T_d}	Trigger Time
i_{T_d}	Trigger Index
n_i	Node Reference Information

List of Figures

List of Tables

Appendix A

GridAgent Details

A.1 GridAgent Events

Within the set of supported input events of the *GA*, a distinction is made between so-called messages, which are used to exchange information between several *GAs* and between an *GA* and its *SPAs*. In addition, the *GA* must support the already known *TM* event for synchronization, which may contain *GA* specific *Triggers*. These set of supported triggers wrapped into a *TM* therefore has to be defined in addition to the general input events. Furthermore, the *GA* supports a number of individual triggers which are not communicated to the *SimS* since they are used exclusively for a state transition within the same tick.

Table A.1: Supported *GridAgent* input events

Acronym	Name	Additional attributes
TM	*TriggerMessage*	T_d, i_{T_d}, d
RGP	*RequestGridPowerMessage*	n_i, i_{sweep}
RSV	*RequestSlackVoltageMessage*	n_i, i_{sweep}
PGP	*ProvideGridPowerMessage*	n_i, S_{n_i}
PSV	*ProvideSlackVoltageMessage*	n_i, U_{n_i}, i_{sweep}
APC	*AssetPowerChangedMessage*	S_{SPA}
APU	*AssetPowerUnchangedMessage*	S_{SPA}

All input events that can be processed by a *GA* are shown in Table A.1 where n_i is a node of the physical grid the power or voltage information is received for and i_{sweep} is the counter for the outer loop of a the backward-forward sweep executed in the case of a coupled overall grid. The set of processable input events X is defined accordingly as:

$$\mathbf{X} = \{TM, RGP, RSV, PGP, PSV, APC, APU\} \tag{A.1}$$

The different supported input trigger that may be wrapped into a TM are given in Table A.2 where is the current logical time tick t_{T_d}.

Table A.2: Supported *GridAgent* input trigger

Acronym	Name	Additional attributes
IT	*InitializeTrigger*	–
AST	*ActivityStartTrigger*	t_{T_d}
$SGST$	*StartGridSimulationTrigger*	t_{T_d}
$PNST$	*PrepareNextSweepTrigger*	t_{T_d}
$FGST$	*FinishGridSimulationTrigger*	t_{T_d}
$DPFT$	*DoPowerFlowTrigger*	t_{T_d}
$CPDT$	*CheckPowerDifferencesTrigger*	t_{T_d}

The different output events supported by a *GA* are depicted in Table A.3 and the output event set is defined accordingly as

$$\begin{aligned}\mathbf{Y} = \{&STM, CM, RAP, RGP, TSV, PGP,\\ &PSV, PNST, FGST, DPFT, CPDT\}\end{aligned} \tag{A.2}$$

Table A.3: Supported *GridAgent* output events

Acronym	Name	Additional attributes
STM	*ScheduleTriggerMessage*	to-be-scheduled trigger & and self reference
CM	*CompletionMessage*	completed trigger id & an optional bag of new-to-be-scheduled triggers
RAP	*RequestAssetPowerMessage*	n_i, S_{n_i}
RGP	*RequestGridPowerMessage*	n_i, i_{sweep}
RSV	*RequestSlackVoltageMessage*	n_i, i_{sweep}
PGP	*ProvideGridPowerMessage*	n_i, S_{n_i}
PSV	*ProvideSlackVoltageMessage*	n_i, U_{n_i}, i_{sweep}
$PNST$	*PrepareNextSweepTrigger*	t_{T_d}
$FGST$	*FinishGridSimulationTrigger*	t_{T_d}
$DPFT$	*DoPowerFlowTrigger*	t_{T_d}
$CPDT$	*CheckPowerDifferencesTrigger*	t_{T_d}

Appendix B

Basic Photovoltaic angles and conventions

To determine the respective proportions of direct, diffuse and reflected radiation, a number of calculations are necessary, which require a basic understanding of the geometric relationships between the sun and a surface relative to the Earth at any given time. To facilitate the understanding of the equations used in Section 4.6, the necessary angles as well as basic conventions for the model calculations based on [141] are described below. Please note that the following explanations are limited to the necessary descriptions. For a comprehensive and exhaustive description of all angles and conventions, please refer to [141]. Unless otherwise specified, all angles used below are indicated in radian (rad).

Day Angle J

The day angle J is the basis for the calculation of the solar declination angle δ as well as the hour angle ω of the respective day. It can be calculated depending the current day of the year n as follows:

$$J(n) = 2\pi \cdot \left(\frac{n-1}{365}\right), n \in \{1, 2, ..., 366\} \tag{B.1}$$

Sun Declination Angle δ

The sun declination angle represents the angular position of the sun at solar noon with respect to the plane of the equator. It can vary between ±23.45° while a positive sign indicates a northern position, a negative sign a southern position. According to [149] it can be calculated as follows:

$$\begin{aligned}\delta = 0.006918 - 0.399912 \cdot \cos J + 0.070257 \cdot \sin J \\ - 0.006758 \cdot \cos 2J + 0.000907 \cdot \sin 2J \\ - 0.002697 \cdot \cos 3J + 0.00148 \cdot \sin 3J\end{aligned} \tag{B.2}$$

Extraterrestrial Radiation I_0

Due to the earth-sun distance and the radiation emitted by the sun, the extraterrestrial radiation available for a PV unit varies in the course of a year. To determine the available extraterrestrial radiation, a day angle J dependant eccentricity correction factor e is multiplied by the solar constant G_{sc} ($\approx 1367\,\mathrm{W/m^2}$). [149]

$$I_0 = e \cdot G_{\mathrm{sc}} \tag{B.3}$$

with

$$\begin{aligned} e = 1.00011 &+ 0.034221 \cdot \cos J + 0.001280 \cdot \sin J \\ &+ 0.000719 \cdot \cos 2J + 0.000077 \cdot \sin 2J \end{aligned}$$

Hour Angle ω

The hour angle ω is a conceptual description of the rotation of the earth around its polar axis. It describes the angular displacement of the sun east or west of the local meridian and changes 15 ° per hour. It starts with negative values in the morning, arrives at 0° at noon (solar time) and goes on with positive values in the afternoon. It can be calculated as follows (taken from [144], [150], [151]):

$$\omega = \left((ST - 12) \cdot 15 \frac{\circ}{\mathrm{h}} \right) \cdot \frac{\pi}{180\,^\circ} \tag{B.4}$$

The required local solar time ST has to be provided in hours and can be calculated as the sum of the local mean time LMT and the equation of time ET as follows

$$ST = LMT + ET. \tag{B.5}$$

The corresponding LMT value can be computed using the angular location east or west of PV unit (longitude) λ in degree ° and the current central eastern time CET by applying

$$LMT = CET - 4 \cdot (15° - \lambda) \tag{B.6}$$

and the corresponding ET value can be calculated using the day angle J in the following equation:

$$\begin{aligned} ET = 0.0066 &+ 7.3525 \cdot \cos(J + 1.4992378274631293) \\ &+ 9.9359 \cdot \cos(2J + 1.9006635554218247) \\ &+ 0.3387 \cdot \cos(3J + 1.8360863730980346). \end{aligned} \tag{B.7}$$

Latitude ϕ

Angular location north or south of the equator (latitude). It starts with a positive value in the north and ends with a negative value in the south. $-90° \leq \phi \leq 90°$

Solar Altitude Angle $\alpha_{\mathbf{s}}$

Represents the angle between the horizontal and the line to the sun, that is, the complement of the zenith angle. [144]

$$\sin \alpha_{\mathrm{s}} = \sin \phi \cdot \sin \delta + \cos \delta \cdot \cos \omega \cdot \cos \phi \tag{B.8}$$

Zenith Angle θ_z

Represents the angle between the vertical and the line to the sun, that is, the angle of incidence of beam radiation on a horizontal surface. [144]

$$\theta_z = \frac{\pi}{2} - |\alpha_s| \tag{B.9}$$

Air Maas AM

Air mass defines the direct optical path length through the atmosphere of the earth. A reasonable good approximation is given by Equation B.10 provided by [152]. The required ratio r_{EA} between the radius of the earth $R_{\mathrm{E}} = 6371\,\mathrm{km}$ and the approximated effective height of the atmosphere $y_{\mathrm{atm}} \approx 9\,\mathrm{km}$ can then be calculated using the formula $r_{EA} = \frac{R_{\mathrm{E}}}{y_{\mathrm{atm}}}$ to approximately $707.8\overline{8}$.

$$AM = \sqrt{(r_{\mathrm{EA}} \cdot \cos \theta_z)^2 + 2r_{\mathrm{EA}} + 1} - r_{\mathrm{EA}} \cdot \cos \theta_z \tag{B.10}$$

Angle of Incidence θ_g

Represents the angle between the beam radiation on a surface, here the PV panel, and the normal to that surface. According to [144] it can be calculated using

$$\begin{aligned}\theta_g = \ & \arccos(\sin\delta \cdot \sin\phi \cdot \cos\gamma_e - \sin\delta \cdot \cos\phi \cdot \sin\gamma_e \cdot \cos\alpha_e \\ & + \cos\delta \cdot \cos\phi \cdot \cos\gamma_e \cdot \cos\omega + \cos\delta \cdot \sin\phi \cdot \sin\gamma_e \cdot \cos\alpha_e \cdot \cos\omega \\ & + \cos\delta \cdot \sin\gamma_e \cdot \sin\alpha_e \cdot \sin\omega).\end{aligned} \tag{B.11}$$

where

δ	= sun declination angle
ω	= hour angle
γ_e	= surface slope angle
α_e	= sun azimuth
ϕ	= latitude of the surface, here PV unit panel

Appendix C

Photovoltaic Model Validation Parameters

In order the validate the functionality of the newly developed PV model, data provided by the RN platform is used. Table C.1 contains the configuration parameters for the used MERRA-2 weather data and Table C.2 contains the model configuration parameters used to parametrize the RN PV model.

The parameters used for the newly developed PV model for its validation are given in Table C.3. For the sampling of different PV unit sites, 100 randomly selected locations within Germany were used which are given in Table C.4.

Table C.1: Renewables Ninja MERRA-2 weather data parameters

Parameter	**Value**
Dataset	MERRA-2 (global)
Year of data	2019

Table C.2: Renewables Ninja PV model configuration parameters

Parameter	**Value**
PV unit capacity	100 kW
System loss as fraction	0.1
Tracking	None
Tilt	35 °
Azimuth	0 °

Table C.3: Developed PV model configuration parameters

Name	**Value**
Rated apparent power	100 kVA
Power factor	0.95
Surroundings Albedo	0.3 °
Inverter efficiency factor	90 %
Azimuth	0 °
Module elevation angle	35 °

Table C.4: Sampled PV unit sites used for validation ($n = 100$)

Latitude	Longitude	Latitude	Longitude
51.66	7.11	52.69	9.78
48.34	7.12	52.07	9.88
53.24	7.14	49.09	9.92
50.44	7.16	49.14	9.96
50.93	7.23	48.07	9.97
50.22	7.31	52.99	9.99
52.01	7.39	48.82	9.99
53.71	7.46	52.21	10.07
51.22	7.47	52.3	10.13
52.73	7.57	51.18	10.15
51.15	7.58	52.35	10.24
53.77	7.63	49.3	10.29
51.27	7.79	52.77	10.29
52.03	7.80	49.18	10.31
48.52	7.81	51.25	10.33
51.48	8.01	49.66	10.50
51.88	8.04	49.09	10.58
52.81	8.07	51.86	10.60
49.67	8.07	50.80	10.68
51.62	8.08	50.54	10.80
52.45	8.09	52.85	10.88
48.93	8.12	50.60	10.91
53.23	8.16	48.58	10.98
52.87	8.20	48.24	11.03
49.60	8.20	50.53	11.21
49.65	8.25	53.85	11.36
49.28	8.38	49.64	11.51
53.49	8.38	52.01	11.62
49.53	8.43	48.19	11.73
53.70	8.50	49.91	11.79
52.78	8.56	51.24	11.90
51.77	8.61	49.26	11.97
50.47	8.62	48.67	12.05
51.01	8.64	53.50	12.10
48.49	8.66	53.47	12.15
53.01	8.69	52.80	12.24
51.41	8.81	50.82	12.25
48.16	8.83	49.37	12.31
48.51	8.98	48.96	12.38
49.15	9.00	50.36	12.44
53.49	9.04	49.31	12.47
51.49	9.08	52.83	12.53
51.76	9.18	53.33	12.62
48.93	9.20	48.96	12.63
53.06	9.20	49.56	12.72
48.99	9.47	49.82	12.75
48.42	9.47	51.72	12.78
52.29	9.51	53.15	12.85
50.75	9.53	48.35	12.88
51.11	9.74	53.43	12.89

Appendix D

Application Example Amendments

D.1 SIMONA-Ecosystem

Performing bottom-up simulations to analyze complex systems often poses a number of challenges due to their complex, detailed and heterogeneous structure. In particular, the parameterization of individual simulation runs contains a high risk of error. With the agent-based structure of SIMONA, which allows individual behavior parameterizations of single system participants, the parameterization becomes even more complex.

For this reason, in addition to the core functionality of SIMONA, a number of tools have been developed that simplify parameterization, application and results analysis (see also Figure 1.3). The resulting SIMONA-Ecosystem supports pre- and post-process operations from data conversion and preparation, through plausibility checking, to evaluation and analysis. It consists of the following components:

PowerSystemDatamodel The PowerSystemDatamodel (PSDM) is a data model particularly developed for bottom-up energy system analysis. It is first of all an object-relational, tabular description of the power grid, different energy system assets as well as their operating characteristics.[19] Additionally, a reference implementation in *Java* which consists of the models, additional input and output capabilities as well as validation tools is available.[20] The PSDM is the main input data format supported by SIMONA.

simBench2psdm The simbench2psdm[21] project is a tool which converts power system models from the SimBench format to PSDM which effectively makes more than 200 SimBench grid models out of the box available to SIMONA.

powerFactory2psdm The powerFactory2psdm[22] project provides functionalities to convert grid models available in the proprietary DIgSILENT PowerFactory to PSDM.

[19] PSDM specification and documentation - https://powersystemdatamodel.readthedocs.io/
[20] PSDM Java reference implementation on GitHub – https://github.com/ie3-institute/PowerSystemDataModel
[21] simbench2psdm on GitHub – https://github.com/ie3-institute/simbench2psdm
[22] powerFactory2psdm on GitHub – https://github.com/ie3-institute/powerFactory2psdm

NetPad++ A graphical representation of grid models in the PSDM format can be generated using NetPad++[23]. The projects main target is not only to draw the electrical grid for visual plausibility checks, but also to support the adaption of existing PSDM grid models. For this purposes already small grid edit functions are available. Additional edit functions may be added on a request base in the future.

DWDWeatherTools With regard to weather data acquisition, the DWDWeatherTools have been developed to collect and transform data provided by the Federal Meteorological Service in Germany (Deutscher Wetterdienst - DWD). It supports the *ICON-EU* [153] data model and converts it to the corresponding PSDM tabular format.

SIMONA-Analysis SIMONA-Analysis provides a variety of supporting functions that allow for the evaluation and analysis of SIMONA simulation results. This part of the SIMONA-Ecosystem is not yet available under an OS license.

In the context of this thesis, a number of contributions have been made to each of these tools. These contributions include tool project initiation, conceptualization and architecture design, as well as concrete implementations. As many of them are available as OS projects, several other persons made and still make contributions to these tools.

[23] NetPad++ on GitHub – https://github.com/ie3-institute/NetPadPlusPlus

D.2 SimBench Grid Database

The SimBench grid database contains a variety of different, freely available benchmark grid models from the low to extra-high voltage. Motivated by the objective to provide a consistent and holistic grid data set for researchers and practitioners, within the corresponding research project[24] a general benchmark grid generation methodology has been developed and presented in [154]. In [12] the approach has been applied to generate representative benchmark grids. The dataset consist of 246 grid models with 41 different distribution grid structures. The models differ in their type, their grid scenario and their specific structure with regard to circuit breaker or switches, respectively. In addition to the actual grid structure, each benchmark grid also includes different system participant time series for load, generation storage and aggregated time series. The latter can be used to substitute adjacent connected downstream grids. Parts of the generation time series were created with the help of a previous version of SIMONA and published in [155].[25] Furthermore, every grid model also contains node voltage magnitude and angle reference results, generated at the nominal operation of all system participants. For additional in-depth details on the grid dataset please refer to [12], [147].

At the time of writing of this thesis, the resulting datasets with the current version number 1 are publicly available on the SimBench website[26]. Each grid in the dataset is available in three different scenarios or expansion stages. The scenario codes are "0 – today", "1 – tomorrow" or "2 – day after tomorrow". Depending on the scenario number, they contain different expansion stages for system participants and grid infrastructure. All grid models can be downloaded either as comma-separated values (csv) files or in the proprietary data formats of the power system simulators DIgSILENT PowerFactory, INTEGRAL or PSS®SINCAL. With the help of the simbench2psdm converter the provided csv files can easily downloaded and converted into PowerSystemDatamodel which makes them usable für SIMONA accordingly.

[24]Simulation Benchmark (SimBench) – Simulationsdatenbasis zum einheitlichen Vergleich von innovativen Lösungen im Bereich der Netzanalyse, Netzplanung und -betriebsführung (eng.: Simulation database for uniform comparison of innovative solutions in the field of grid analysis, planning and operation); funded by the Federal Ministry for Economic Affairs and Energy under the grant number 0325917.

[25]The PV time series have been generated using the model presented in Section 4.6.

[26]SimBench dataset website - https://simbench.de/datensaetze/

D.3 SimBench Validation Information

Table D.1: SimBench benchmark grid model codes used for the validation of SIMONA

SimBench Code
1-LV-rural1–0-no_sw
1-LV-rural1–1-no_sw
1-LV-rural1–2-no_sw
1-LV-rural2–0-no_sw
1-LV-rural2–1-no_sw
1-LV-rural2–2-no_sw
1-LV-rural3–0-no_sw
1-LV-rural3–1-no_sw
1-LV-rural3–2-no_sw
1-LV-semiurb4–0-no_sw
1-LV-semiurb4–1-no_sw
1-LV-semiurb4–2-no_sw
1-LV-semiurb5–0-no_sw
1-LV-semiurb5–1-no_sw
1-LV-semiurb5–2-no_sw
1-LV-urban6–0-no_sw
1-LV-urban6–1-no_sw
1-LV-urban6–2-no_sw
1-MV-comm–0-no_sw
1-MV-comm–1-no_sw
1-MV-comm–2-no_sw
1-MV-rural–0-no_sw
1-MV-rural–1-no_sw
1-MV-rural–2-no_sw
1-MV-semiurb–0-no_sw
1-MV-semiurb–1-no_sw
1-MV-semiurb–2-no_sw
1-MV-urban–0-no_sw
1-MV-urban–1-no_sw
1-MV-urban–2-no_sw
1-MVLV-comm-3.403-0-no_sw
1-MVLV-comm-3.403-1-no_sw
1-MVLV-comm-3.403-2-no_sw
1-MVLV-comm-4.416-0-no_sw
1-MVLV-comm-4.416-1-no_sw
1-MVLV-comm-4.416-2-no_sw
1-MVLV-comm-5.401-0-no_sw
1-MVLV-comm-5.401-1-no_sw
1-MVLV-comm-5.401-2-no_sw
1-MVLV-comm-all-0-no_sw
1-MVLV-comm-all-1-no_sw
1-MVLV-comm-all-2-no_sw
1-MVLV-rural-1.108-0-no_sw
1-MVLV-rural-1.108-1-no_sw
1-MVLV-rural-1.108-2-no_sw
1-MVLV-rural-2.107-0-no_sw
1-MVLV-rural-2.107-1-no_sw
1-MVLV-rural-2.107-2-no_sw
1-MVLV-rural-4.101-0-no_sw
1-MVLV-rural-4.101-1-no_sw
1-MVLV-rural-4.101-2-no_sw
1-MVLV-rural-all-0-no_sw
1-MVLV-rural-all-1-no_sw
1-MVLV-rural-all-2-no_sw
1-MVLV-semiurb-3.202-0-no_sw
1-MVLV-semiurb-3.202-1-no_sw
1-MVLV-semiurb-3.202-2-no_sw
1-MVLV-semiurb-4.201-0-no_sw
1-MVLV-semiurb-4.201-1-no_sw
1-MVLV-semiurb-4.201-2-no_sw
1-MVLV-semiurb-5.220-0-no_sw
1-MVLV-semiurb-5.220-1-no_sw
1-MVLV-semiurb-5.220-2-no_sw
1-MVLV-semiurb-all-0-no_sw
1-MVLV-semiurb-all-1-no_sw
1-MVLV-semiurb-all-2-no_sw
1-MVLV-urban-5.303-0-no_sw
1-MVLV-urban-5.303-1-no_sw
1-MVLV-urban-5.303-2-no_sw
1-MVLV-urban-6.305-0-no_sw
1-MVLV-urban-6.305-1-no_sw
1-MVLV-urban-6.305-2-no_sw
1-MVLV-urban-6.309-0-no_sw
1-MVLV-urban-6.309-1-no_sw
1-MVLV-urban-6.309-2-no_sw
1-MVLV-urban-all-0-no_sw
1-MVLV-urban-all-1-no_sw
1-MVLV-urban-all-2-no_sw

Table D.2: Maximum upper and lower voltage magnitude and angle deviation between the results of SIMONA and the provided SimBench reference results

	Δv_{mag} in p.u.		$\Delta\delta$ in °	
	lower	upper	lower	upper
LV	$-6.70 \cdot 10^{-4}$	$9.20 \cdot 10^{-3}$	$-0.047, 41$	$0.488, 18$
MV	$-1.00 \cdot 10^{-5}$	$3.50 \cdot 10^{-4}$	$-0.019, 99$	$0.000, 34$
MV/LV	$-8.90 \cdot 10^{-4}$	$1.15 \cdot 10^{-2}$	$-0.073, 55$	$0.629, 79$

D.4 Application Example Properties

Table D.3: Properties of SimBench model “1-MVLV-rural-4.101-0-no_sw”

Property	**Value**
Overall Number of Nodes	146
HS	1
MS	102
NS	43
Total Circuit Length	116.7358 km
Installed Load $S_{\mathrm{r,L}}$	18,554.83 kVA
Installed Feed-in $S_{\mathrm{r,G}}$	25,564.98 kVA
Downstream LV	12,513.50 kVA
Wind MV	11,400 kVA
PV MV	326.48 kVA
Hydro MV	385.0 kVA
Biomass MV	940.0 kVA
Feed-In-to-Load-Ratio	57.97 % / 42.05 %

Table D.4: Assumed PV park model properties for each expansion stage

Parameter	**Symbol**	**Value**
Rated Apparent Power	S_{r}	2000 MVA
Nominal Power Factor	$\cos(\varphi)$	0.95
Latitude	ϕ	53.66°
Longitude	λ	11.41°
Surroundings Albedo	ρ	0.20°
Inverter Efficiency Factor	η	96 %
Azimuth	α_e	−11.46°
Module Elevation Angle	γ_e	33.63°

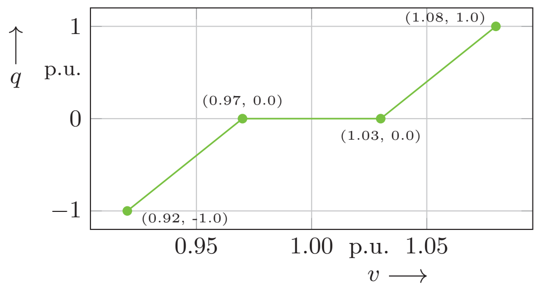


Figure D.1: Applied $Q(V)$ control characteristic in the PV park expansion stage

Evidence of Scientific Activity

Scientific Publications

[P1] J. Hiry, J. von Haebler, U. Häger, C. Rehtanz, G. Blanco, and A. Martinez, "Agent-based modelling of cost efficient and stable transmission grid expansion planning," in *Highlights of Practical Applications of Scalable Multi-Agent Systems. The PAAMS Collection - International Workshops of PAAMS 2016, Sevilla, Spain, June 1-3, 2016. Proceedings*, J. Bajo, M. J. Escalona, S. Giroux, P. Hoffa-Dabrowska, V. Julián, P. Novais, N. S. Pi, R. Unland, and R. A. Silveira, Eds., ser. Communications in Computer and Information Science, vol. 616, Springer, 2016, pp. 356–368. DOI: 10.1007/978-3-319-39387-2_30.

[P2] J. Hiry, C. Kittl, Z. Hagemann, and C. Rehtanz, "Das Potential spannungsebenenübergreifender Zeitreihensimulationen für die Verteilnetzplanung," in *Smart Energy Konferenz*, U. Großmann, I. Kunold, and C. Engels, Eds., Dortmund: vwh Verlag Werner Hülsbusch, 2017, pp. 133–142, ISBN: 978-3-86488-125-1.

[P3] L. Jendernalik and D. Giavarra and C. Engels and J. Hiry and C. Kittl and C. Rehtanz, "Holistic network planning approach: Enhancement of the grid expansion using the flexibility of network participants," *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 2312–2315, 2017, ISSN: 2515-0855. DOI: 10.1049/oap-cired.2017.0061.

[P4] C. Römer, J. Hiry, C. Kittl, T. Liebig, and C. Rehtanz, "Charging control of electric vehicles using contextual bandits considering the electrical distribution grid," in *KNOWMe: 2nd International Workshop on Knowledge Discovery from Mobility and Transportation Systems*, Dublin, 2018.

[P5] J. Hiry, C. Kittl, and C. Römer and C. Rehtanz and L. Willmes and S. Schimmeyer, "Automated time series based grid extensions planning using a coupled agent based simulation and genetic algorithm approach," in *Proceedings of the 25th International Conference on Electricity Distribution (CIRED)*, 2019. DOI: http://dx.doi.org/10.34890/521.

[P6] J. Hiry, "Wie gelingt die Netzintegration von Elektromobilität?" *ETG Journal*, pp. 40–41, 2019.

[P7] C. Kittl, J. Hiry, C. Pfeiffer, C. Rehtanz, and C. Engels, “Large scale agent based simulation of distribution grid loading and its practical application,” in *Proceedings of the 25th International Conference on Electricity Distribution (CIRED)*, 2019.

[P8] S. Balduin, E. M. Veith, A. Berezin, S. Lehnhoff, T. Oberließen, C. Kittl, J. Hiry, C. Rehtanz, G. Torres-Villarreal, S. Leksawat, and M.-A. Frankenbach, “Towards a universally applicable neural state estimation through transfer learning,” in *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, 2021.

[P9] J. Hiry, J. Peper, S. Peter, C. Kittl, and C. Rehtanz, “Regional spatial distribution of electric vehicles - a Data-Driven approach for distribution grid impact case studies,” in *ETG-Kongress 2021 - Von Komponenten bis zum Gesamtsystem für die Energiewende (ETG-Kongress 2021)*, Wuppertal, Germany, 2021.

[P10] T. M. Schwarz, J. Maasmann, J. Hiry, and C. Rehtanz, “Load prediction tool for ev charging infrastructure,” in *Proceedings of the 26th International Conference on Electricity Distribution (CIRED)*, 2021.

Technical Reports and Committee Work

[R1] Forum Netztechnik/Netzbetrieb im VDE (FNN), "Zielbild Steuerbarkeit von Ladeinfrastruktur für E-Fahrzeuge," Forum Netztechnik/Netzbetrieb im VDE (FNN), Tech. Rep., 2021.

[R2] D. Vangulick, M. Black, S. Deters, R. Girard, L. Held, J. Hiry, V. Jelenecky, S. Koch, D. Kouba, U. Krisper, T. Moya, M. Petit, B. Karima, K. Rudion, M. Uhrig, E. Voumvoulakis, and T. Wieland, "Load modelling and distribution planning in the era of electric mobility," International Conference on Electricity Distribution (CIRED), Tech. Rep., 2021.

Supervised Master Theses

[M1] J. Brechmann, “Entwicklung einer Methodik zur Optimierung des Asset Managements durch Auswertung von Mess- und Betriebsdaten in Windparks,” Supervisors: J. Hiry, Z. Hagemann, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[M2] S. Hintzen, “Konzeptionierung und Implementierung eines agentenbasierten Marktmodells unter Berücksichtigung des konventionellen Kraftwerksparks,” Supervisors: J. Hiry, C. Kittl, Z. Hagemann, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[M3] C. Ruether, “Entwicklung und Bewertung verschiedener Algorithmen zur Spitzenkappung in der Verteilnetzplanung,” Supervisors: C. Wagner, J. Hiry, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[M4] J.-H. Brumund, “Zeitreihenbasierte Identifikation auslegungsrelevanter Nutzungsfälle des Verteilnetzes unter Berücksichtigung negativer sowie positiver Einflussfaktoren,” Supervisors: V. Peters (Westnetz GmbH), C. Kittl, J. Hiry, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[M5] M. Kloß, “Modellierung und Analyse der Wechselwirkung von virtuellen Kraftwerken mit dem elektrischen Verteilnetz,” Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[M6] J. Peper, “Agentenbasierte Modellierung und Simulation der räumlichen und zeitlichen Verteilung von Elektrofahrzeugen,” Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[M7] C. Pfeiffer, “Entwicklung und Evaluation von Verteilungsstrategien für die Lastflussrechnung,” Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Faculty of Computer Science, Chair IV - Practical Computer Science, TU Dortmund University, 2018.

[M8] C. Römer, "Ladesteuerung von Elektrofahrzeugen mit kontextsensitiven Banditen unter Berücksichtigung des elektrischen Verteilnetzes," Supervisors: T. Liebig, J. Hiry, C. Kittl, Unpublished Master Thesis, Faculty of Computer Science, Chair VIII - Artificial Intelligence, TU Dortmund University, 2018.

[M9] T. Stenzel, "Entwicklung und Implementierung von innovativen Abregelungsmechanismen zur Netzintegration von Elektrofahrzeugen," Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[M10] M. Meschede, "Analyse und Erweiterung einer Methodik zur Abreglung von Ladeinfrastruktur zur Netzintegration von Elektrofahrzeugen," Supervisors: J. Hiry, C. Kittl, T. Liebig, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2019.

[M11] T. Steinwachs, "Analyse und agentenbasierte Modellierung von Speichertechnologien zur großflächigen Integration von Elektrofahrzeugen in elektrische Verteilnetze," Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2019.

[M12] M. Wiemann, "Untersuchung der Eignung von Distributed-Ledger-Technologien zum Aufbau regionaler Märkte im elektrischen Verteilnetz," Supervisors: J. Hiry, M. Klaes, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2019.

[M13] T. Oberließen, "Entwicklung eines Modells zur Synthetisierung von Energiebedarfszeitreihen zur Identifikation des zeitliche und räumlich differenzierten Ladebedarfs von Elektrofahrzeugen," Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2020.

[M14] S. Peter, "Entwicklung und Implementierung einer verteilten Scheduling-Strategie für eine ereignisbasierte Verteilnetzenergiesystemsimulation," Supervisors: J. Hiry, C. Kittl, Unpublished Master Thesis, Faculty of Computer Science, Chair IV - Practical Computer Science, TU Dortmund University, 2021.

[M15] N. Steffan, “Development and implementation of an agent-based model to assess approaches that efficiently account stochastic charging needs of battery-electric vehicles,” Supervisors: C. Kittl, J. Hiry, Unpublished Master Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2021.

Supervised Bachelor Theses

[B1] P. Kassack, “Bewertung von Verfahren zur Ersatzwertbildung für die Berechnung von Netzzuständen im Verteilnetz,” Supervisors: J. Hiry, A. Brüggemann, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2016.

[B2] J. Gabrielski, “Analyse und Vergleich von Berechnungsmethoden für eine realitätsnahe Berechnung erzeugter Energie aus Sonneneinstrahlung,” Supervisors: Z. Hagemann, J. Hiry, C. Kittl, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B3] T. F. Grunert, “Identifikation der thermischen und elektrischen Versorgungsaufgabe in städtischen Quartieren anhand von öffentlich verfügbarem Kartenmaterial,” Supervisors: J. Hinker, J. Hiry, C. Kittl, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B4] L. Lenz, “Analyse und Modellierung des Betriebsverhaltens von Nano-/ Mini-Blockheizkraftwerken zur agentenbasierten Netzausbauplanung,” Supervisors: J. Hiry, Z. Hagemann, C. Kittl, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B5] P. Lenz, “Analyse der Anwendbarkeit künstlicher Intelligenz zur Verbesserung der Funktionsweise von regelbaren Ortsnetzstationen,” Supervisors: J. Hiry, C. Kittl, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B6] L. Maaß, “Agentenbasierte Modellierung von Großkraftwerken unter Beachtung technologiespezifischer Randbedingungen und individueller ökonomischer Entscheidungsfindung,” Supervisors: C. Kittl, J. Hiry, Z. Hagemann, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B7] J. Peper, “Analyse und Modellierung des Betriebsverhaltens eines Biomassekraftwerkes zur agentenbasierten Netzausbauplanung,” Supervisors: J. Hiry, Z. Hagemann, C. Kittl, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2017.

[B8] G. Düssel, “Entwicklung eines brown-field-planning-Ansatzes für Niederspannungsnetze unter Nutzung öffentlich verfügbarer Daten des OpenStreetMap-Projekts,” Supervisors: C. Kittl, J. Hiry, D. Sarajlić, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[B9] T. Jäkel, “Vergleichende Analyse von konventioneller und innovativer Messtechnologie zur Detektion von Gaslecks im Verteilnetz,” Supervisors: R. Bachmann (Westnetz GmbH), C. Kittl, J. Hiry, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[B10] C. Nerowski, “Analysing and Modelling Concepts of Virtual Entities for later use in Agent Based Grid Expansion Planning approaches,” Supervisors: C. Kittl, J. Hiry, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[B11] T. Unterluggauer, “Markthochlaufszenarien von Elektrofahrzeugen im ländlichen und (sub-)urbanen Raum,” Supervisors: J. Hiry, G. Göhler (Fraunhofer IAO Stuttgart), Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2018.

[B12] C. Mahr, “Entwicklung einer grafischen Benutzeroberfläche zur Darstellung und Bearbeitung modellierter elektrischer Energienetze,” Supervisors: M. Greve (ef.Ruhr GmbH), J. Hiry, C. Kittl, Unpublished Bachelor Thesis, Hochschule Ruhr West - University of AppliedSciences, 2019.

[B13] A. Ostrop, “Entwicklung eines Greedy-Algorithmus zur Generierung elektrischer Niederspannungsnetze auf Basis öffentlicher verfügbarer Daten,” Supervisors: E. Schubert, J. Hiry, C. Kittl, Unpublished Bachelor Thesis, Faculty of Computer Science, Chair VIII - Artificial Intelligence, TU Dortmund University, 2019.

[B14] N. Steffan, “Entwicklung einer Methodik zur quasi-dynamischen Modellierung möglicher Betriebspunktänderungen flexibler Verbraucher in einer agentenbasierten Netzsimulation,” Supervisors: C. Kittl, B. Matthes, J. Hiry, Unpublished Bachelor Thesis, Institute of Energy Systems, Energy Efficiency and Energy Economics, TU Dortmund University, 2019.

[B15] M. E. Krause, “Einführung moderner Datenbanken in ein agentenbasiertes Simulationssystem für Smartgrids zur Verbesserung der Performanz,” Supervisors: I. Saatz, C. Kittl, J. Hiry, Unpublished Bachelor Thesis, Department of Computer Science, University of Applied Sciences and Arts Dortmund, 2021.

[B16] V. Zachopoulos, “Konzeptionierung und Implementierung einer Kommunikationsplattform zum Einsatz von SIMONA als “Simulation as a Service“,” Supervisors: T. Oberließen, C. Kittl, J. Hiry, Unpublished Bachelor Thesis, Faculty of Computer Science, Chair IV - Practical Computer Science, TU Dortmund University, 2021.

Supervised Study Projects

[S1] M. E. Krause, “Einführung moderner Datenbanken in ein agentenbasiertes Simulationssystem für Smartgrids zur Verbesserung der Performanz,” Supervisors: I. Saatz, C. Kittl, J. Hiry, Unpublished Study Project Report, Department of Computer Science, University of Applied Sciences and Arts Dortmund, 2020.