A generic hierarchic architecture for the coordination of energy management systems.

DENIS BYTSCHKOW

Institut für Informatik
der Technischen Universität München

A generic hierarchic architecture for the coordination of energy management systems

Denis Bytschkow

Vollständiger Abdruck der von der Fakultätät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Uwe Baumgarten
Prüfer der Dissertation:		
	1.	Prof. Dr. Dr. h.c. Manfred Broy
	2.	Prof. Dr. Thomas Hamacher

Die Dissertation wurde am 22.04.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.08.2020 angenommen.

Abstract

Technically, the operation of smart energy systems (SES) is based on various energy management systems (EMS) that pertain different spatial and temporal scales. Despite their technical variety, the responsibilities of EMS are similar. They monitor and control available equipment for power production, consumption or storage to achieve a stable, reliable, sustainable and economic supply of energy. Nowadays, with the penetration of renewable energies accompanied with increasing investments in storage systems, electric vehicles, and systems for heat, such as heat pumps, and combined heat and power units, EMS greatly rise in importance due to their pure number and their impact on our power supply system. The integration of EMS into the overall power supply system becomes inevitable. Integration is a question related to the system architecture. The focus on EMS with similar functionality and their local interdependence lead to the question, whether EMS of buildings should be interconnected with EMS local quarters, which again can be interconnected with EMS of districts, of cities and so on. A hierarchic, more generic system of EMS would appear. The idea is not new, it has already been discussed by Moslehi and Kumar (2010); Grijalva and Tariq (2011); Benz et al. (2015); Howell et al. (2017) and others. However, so far, all discussions introduce such interconnections only as high-level descriptions. They lack concrete specifications of the system's interfaces, the system's behaviour and their relation to mathematical models for control and optimizations that are available for coordination. Therefore, it is still an open architectural question, how to conceptualize the desired hierarchic composition. To demonstrate the usefulness of the architecture it is also necessary to study the effects of such interconnections to the power supply system.

This thesis addresses those open questions and provides several contributions by combining approaches from power system research with software engineering. The essential basis for power system operation is given by mathematical models for optimization and control. The required data for the interconnections and therefore for the specifications of interfaces are implicitly available within those models. We show methodologically how to use them and how they can be composed and decomposed for hierarchical systems, as expected with our architecture. The specification of the generic architecture for a hierarchic EMS composition relies on sound architecture design patterns. We present a constructive approach starting with requirements, the selection of suitable architectural design patterns to meet those and their combination to develop the architecture. The approach is complemented with a technical framework to test such generic architectures in SES context. Finally, we apply the developed solution in two complementary SES case studies to demonstrate the feasibility, the coordination possibilities and the expected impact in terms of power system improvement related to CO_2 emissions and economic costs.

Acknowledgements

This work would not have been possible without freedom I was given to pursue this research topic and the support by various people. In particular, I want to express my deep gratitude for Prof. Manfred Broy for giving me the possibility to work under his supervision on this interdisciplinary topic, supporting me with so many constructive discussions and reviews during the different stages of the work. It was always a pleasure to experience such an open-minded research and guidance. I also thank Prof. Thomas Hamacher for being the second supervisor and his support that combined our two groups of software engineers and power system engineers.

During the last years, I had the luck to work with many great colleagues, here at fortiss, TUM, but also with our industrial partners conducting our combined research activities. I would like to thank all of them. I would like to especially thank Markus Duchon, for his support, the pleasant discussions and his feedback on our research, Bernhard Schätz, who I will always remember as one of the most inspiring and open minded persons I've ever met, Maximilian Junker for his structured, helpful and constructive feedback and Pragya Gupta, for being my roommate for many years. Further, I would like to thank all the colleagues I've worked with, in no particular order, Klaus Becker, Konstantin Schorp, Maximilian Irlbeck, Vasileios Koutsoumpas, Georg Hackenberg, Bianka Dachwald, Eleni Nikolaou-Weiss, Harald Ruess, Jan Mayer, Arpit Bajpai, Venky Pampana, Michael Bernard and Sabine Teufl, but also all the other group members at fortiss.

Working closely with the industry and TUM, I would like to thank everybody for the pleasant work with different research domains, particularly all the people working in the energy (management) domain, including Alexander Vilbig, Niklas Nohlen, Matthias Jeschke, Behnam Tajedini from the Munich Utility, Claudius Reiser, Ralf Nebel from Sauter-Cumulus GmbH, Michael Kramer, Alexandre Capone, Thomas Licklederer from the MSE research group and all the other researcher with whom we collaborated during the last years.

Finally, but most importantly, I would like to thank my lovely wife Natalia, my two little sons Roman and Daniel, my parents and my brother Witali. The support during the last years was remarkable. Thank you.

Contents

1 | Introduction

Today, more then ever, we know that we are confronted with an ecological crisis. An ecological crisis that can only be solved, if we reverse the emission of greenhouse gases as much and as quickly as possible. The transition of our energy supply system towards a more sustainable network of power system components is of paramount importance for the required solution. This includes more renewable energy production of different types that are synchronized with the energy demand, storage and flexibilities. Managing and controlling this variety of components and systems is essential for a reliable and stable power system operation. This coordination involves Information and Communication Technologies (ICT) in power systems throughout the different locations and system scales and often denoted as smart energy systems (SES). The work on SES architectures for coordination and balancing is the topic of this thesis.

This chapter starts with the motivation for the topic in Section 1.1. Afterwards, Section 1.2 presents a number of open problems in this field. Section 1.3 discusses the research objectives of our work and describes the contributions in a brief summary. Finally, Section 1.4 gives an outline of this thesis and presents the previously published material.

1.1 Motivation

An increasing share of renewable energy production is essential to achieve a power system that is more sustainable and independent of fossil fuels. Renewable energy sources convert the power from naturally available conditions and are therefore inherently fluctuating and decentralised. Due to the volatility and an increased uncertainty, a larger share of renewable energies demands for more control capabilities to balance the power supply and demand at every single point of time. The necessity is inevitable, as demonstrated by a significant energy curtailment increase in both aspects, frequency and volume, during the last years due to an increasing share of renewable energies (Schermeyer, 2018).

Today, renewable energy sources are complemented with technologies such as storage components, charging stations for electric vehicles and devices for heating and cooling such as combined heat and power (CHP) units or heat pumps. Their integration leads to energy management systems (EMS). These are software systems for management and automation of energy equipment. They range from simple monitoring and control systems for single devices, to rule-based automation systems that adapt for instance the charging and discharging rate of storages based on current production and demand, up to integrated complex systems that combine the heating, cooling, electricity and mobility domain using sets of automation rules or even model based control to optimize the internal energy usage. EMS are able to change the behaviour of local components with respect to power consumption and production by adapting the schedules for storage, production or demand (if available). Hence, they offer additional options to make the system more flexible reducing the necessity for energy curtailment. Because the majority of the recent renewable energy sources, storages, charging stations are installed at the distribution network or even at the building level, the control capabilities shift from large systems at the top voltage levels, to smaller systems at lower voltage levels. This challenges the current concepts for centrally organized power systems, as decision making for control moves towards lower system levels.

To increase the share of renewable energy sources and reduce energy curtailment EMS should be coordinated. It means that they have to be connected to some kind of coordination system that monitors them and sends coordination signals to guide their behaviour. Coordination of EMS (i.e. interactive control systems) combines two areas of research. First, the coordination of power systems is essentially based on mathematical models, which model the system to derive schedules for operation and use flexibilities when deviations are observed. Second, in order to perform calculations based on such models, a coordination system requires to collect data from and send data to the connected EMS. This an integration effort, which particularly asks for the right system interfaces and system behaviour of the local EMS and coordination systems.

Further, our energy supply system is hierarchical with respect to the organization and responsibilities of the network operators and the technical infrastructure with its hierarchical voltage levels. It is therefore an open question if the coordination of EMS (and also their integration) should be hierarchic as well. The idea is that several EMS are combined into one group, for instance a group for one particular area. The coordination system of that area coordinates the available EMS to improve for instance costs or carbon emissions. In addition, the coordination system aggregates the control capabilities and offers them as flexibilities to the next hierarchical level. In consequence, the coordination system represents a group that is part of a larger group, which is again treated as a coordination system with its own responsibilities. All those coordination systems have the role of an EMS creating a hierarchy as presented in Figure 1.1.

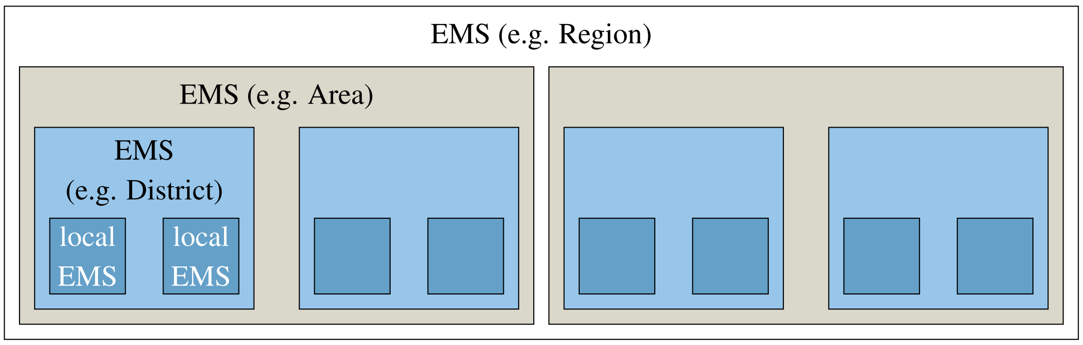


Figure 1.1: Hierarchic composition of energy management systems.

Note, the internal composition of EMS is allowed to be different, since each EMS has a different set of internal devices. For instance, an EMS at the building level has its own set of renewable energy sources, storage components, etc. EMS that represent districts might differ with the availability of district heating capabilities. One EMS might have an infrastructure that operates at high temperature levels, while another one has lower temperature levels and a third one does not have it at all. Also local transmission constraints might be different. Nevertheless, each EMS adds control capabilities to the overall system and this is essential to increase the share of renewable energies. The hierarchical integration adds explicit system boundaries. These are important for two reasons. First, system boundaries allow to add internal rules or metrics for each EMS. They help to specify internal operation goals or add local safety constraints for devices. Second, system boundaries help with abstraction and abstraction helps to find more generalizable interfaces, which is crucial for system integration. However, the hierarchy and EMS system boundaries open a number of related problems that need to be solved.

1.2 Problem Statement

Today, we see already a number of integration efforts to create coordination systems. Most of the efforts are focused on the device level. For instance, small scale, decentralized energy resources with a low and medium generation are aggregated into Virtual Power Plants (VPP) to substitute the capabilities of large power plants and participate at the energy market as introduced by Willems (2005). Devices are also integrated within local networks to support the local power supply. This is known as microgrids as introduced in Lasseter and Piagi (2004). Beyond the pure substitution of bulk power plants and local optimizations, the SES vision introduces new applications to increase the flexibility and control capabilities within our power supply system, such as prosumers[1], demand side management, charging stations for electric vehicles or flexibilities. The ultimate goal is the creation of a flexible and reliable network of interacting power system components that provide a stable, reliable, sustainable and economic supply of energy. Therefore, we need to lift the integration from single devices to more complex EMS.

Problem 1: Integration with current technologies Current technological solutions in the energy domain that integrate controllable components are based on the idea that human operators have systems to supervise a certain area and intervene when something happens. For this task Supervisory Control and Data Acquisition (SCADA) systems have been developed. A SCADA system consists of software and hardware components. Its main purpose is to collect information from remote locations, transfer data to a central facility, and provide graphical or textual representation for operators that monitor and control the entire system. Since, the infrastructure is essential for a stable power supply, SCADA systems requires a high degree of availability, reliability and integrity. Availability and reliability are addressed by SCADA architectures with redundancy. The integrity requirement is addressed by encapsulating the system from external access as much as possible. This includes physical encapsulation of control rooms with dedicated access control, private communication networks, demilitarised zones for data access, virtual private networks (VPN) and further similar approaches. A good overview about SCADA fundamentals is described in the NIST analysis by Stouffer et al. (2006), where he introduces the common architecture decisions for SCADA. Today, most companies in the utility and distribution sector use SCADA systems that are enriched with specific applications from the energy domain.

To integrate EMS using SCADA technology means that several severe problems need to be solved. First, the integration of external components (i.e. from another manufacturer), particularly end-user equipment is problematic. It is heterogeneous and not that secure as the clearly defined and highly secured industrial infrastructure equipment that is currently integrated into SCADA systems. Changes of end-user equipment appear also uncoordinated and more often. Consequently, system updates are required more frequently and maintainability efforts increase, but that is time and cost intensive Barnes et al. (2004). Second, end-user equipment has a higher risk of failures. In current SCADA systems, failures (e.g. overloads of physical connections) trigger alarms on which engineers react with appropriate actions. Such human interaction is

[1]The term prosumer describes a system that can produce or consume power. Storage systems can be part of a prosumer as well.

only possible if the alarm frequency is low. By integrating lower voltage levels and thousands of components, the alarm frequency might increase drastically and the necessity for human interaction rises significantly. Third, SCADA systems operate in encapsulated environments, as much as possible. Secure operation is provided by controlled system access. Connections are only established to clearly defined components, which are often physically locked from public access. The integration of new, non-locked end-user equipment introduces severe threats. It is therefore a core question, how end-user equipment can be integrated, without opening the access to critical components of the infrastructure.

Problem 2: Ambiguous architecture specifications For the realization of the smart energy vision, work on the architecture is as important as work on technologies and products. The idea of aggregation and hierarchies is not completely new. It has already been intensively discussed under different terms in the literature. Commonly appearing terms are 'Virtual Power Plants', 'Microgrids' and 'Aggregators' (Asmus, 2010; Hashmi et al., 2011). Also terms like 'Virtual Utilities', 'Active Distribution Networks', 'Prosumers' or 'Cells' are used for this context (Lund et al., 2005; Braun and Strauss, 2008; Grijalva and Tariq, 2011; Benz et al., 2015). Previous work on SES architectures, in particular those with the focus on new organization schemes, such as the work of Grijalva and Tariq (2011) on prosumer oriented architectures; the work on aggregation principles, e.g. (Asmus, 2010; Braun and Strauss, 2008); or the work on a cellular energy supply system (Benz et al., 2015) introduce the concept of hierarchy and EMS integration on a rather informal level. This causes many problems for the systems developer, such as ambiguity, wrong conclusions and difficulty when checking the conformance of a system to the suggested architecture. For example, the work by Benz et al. (2015) concentrates on the demonstration that cells, which represent a group of SES components, can be controlled in such way, that they provide flexibilities to adapt power production and demand. However, he gives no information about the involved control systems, their interfaces or the communicated data. Nevertheless, the idea of a hierarchical integration of similar systems remains appealing and is currently being discussed as the next architecture (Reuter and Breker, 2018). The ideas are supported with great funding[2]. However, so far we have not seen concrete architecture specifications.

Such vague descriptions are problematic when they claim that a system benefits from a hierarchic architecture but give no information about the concrete system architecture leaving room for multiple, miscellaneous interpretations. In an architecture specification, it should be very clear, which systems are available, what is their system boundary, which data is offered by which system, which interfaces exist, which functionality exist and which system executes which functionalities. For instance, it should be clear, which system executes the core coordination calculations like planning and control of the power components, how it receives data and how it sends data. Basically we need a specification with three concrete ingredients to establish a hierarchic coordination for EMS. These include coordination models that define the data that is required at the interfaces, descriptions of the interfaces based on that model, and the system behaviour specification that describes the data processing for the integrated EMS.

[2]For instance the SINTEG initiative, with a total budget of 500 Mio. € (Sinteg, 2018), and large projects within this initiative, for instance C/sells, where cells are the fundamental architectural entity (Reuter and Breker, 2018)

Problem 3: Specification of EMS system boundaries, interfaces and coordination models A number of SES applications already integrate and control small scale renewable power system components. A popular example is a VPP. It combines (simple) EMS from different locations to a system (mostly a SCADA system), in order to provide the same generation capabilities as a bulk power plant. This enables access for smaller units to the same energy market as large power plants. This is a good *aggregation* example. Another popular SES application is the combination for control systems in order to coordinate a set of local components in a certain area. This is the so-called microgrid. It combines various EMS components for a better balancing of the local consumption and generation. The regional coordination has multiple benefits. It allows to optimize local energy resources. It reduces the strain on the network and saves grid extension costs. This is good *coordination* example.

A holistic architecture of a coordination system should allow a combination of those two concepts. But today, this is still not possible due to a number of problems. Currently, both systems have direct access to control the power system components. In a combined system, however, we need a clear separation into one local EMS, which has the direct control of components and considers a number of local constraints, and one EMS for aggregation and coordination, which collects the available resources and transfers them into a larger scale to consider synergies from the network and remote components. For that, clear system boundaries with clear interfaces and models for coordination are necessary. To understand the impact of such hierarchies, we need to specify how the systems shall interact and evaluate those applications in a running system. Otherwise, the architecture specifications remains incomplete, since they might miss important details for the desired coordination systems.

Problem 4: Evaluation of hierarchic architectures for EMS coordination Once an architecture for a system is established, it must be tested and evaluated with respect to its technical feasibility and whether the benefits obtained are in line with the expected results. For testing and evaluation of SES architectures that integrate EMS so far no test environment is available. The reason is simple. Today ,there are hardly open EMS that provide services and can be integrated to test their behaviour. Most of the research focuses on concepts and research prototypes. To evaluate architectures for the coordination of EMS we require therefore a technical environment that is capable to create different EMS or to integrate available EMS for further analysis.

1.3 Research Objectives and Contributions

In this thesis, the goal is to increase the understanding for the coordination of EMS from the architectural point of view. In that context we also particularly investigate hierarchic architectures to clarify what such architectures mean for compositions and which benefits (and drawbacks) we might expect from them. Therefore, we formulate several research objectives that we approach in the following of this thesis.

Research objective 1: *Determine the essential specification elements for the hierarchic composition of systems to achieve a hierarchic coordination of energy systems.*

The first research objective targets the concept of system composition for energy systems into groups which are hierarchically structured. The goal is to clarify the specification elements and prerequisites to create groups of EMS and investigate how those groups interact internally and towards higher levels. When we investigate the composition of systems, the notion of a system boundary and system behaviour is important. Therefore, we need to understand what are the system boundaries of coordination systems from the energy modelling perspective and software perspective. Are they the same or do we see significant differences so that we need to extend our specifications in order to develop such coordination systems. What is the behaviour that we observe in hierarchical systems? Further, for the technical implications we need to understand, if hierarchical coordination systems are technically feasible. We need also to find out if and how a hierarchical architecture limits the technical implementation, to understand if the development of additional protocols is required and how available technical concepts and best practises, such as architecture design patterns, help to design hierarchic systems.

Research objective 2: *Develop a coordination approach for EMS based on mathematical models using the specification elements of objective one.*

Coordination approaches for EMS are based on mathematical models. The second research objective is to examine the coordination of the EMS starting from general mathematical approaches and transferring them into the topic of hierarchic EMS compositions. The challenge here is to understand how energy models can be transferred to interactive systems with clear interfaces embedded in hierarchical structures. We address here the following questions: Should energy models be adapted or represented in a different way, as soon as systems architectures are involved? Which interfaces are suitable to transfer the data required by those models? How do coordination approaches affect the behaviour of systems and their input and output interfaces? The aim is to investigate synergies from energy models research with the development process of coordination systems.

Research objective 3: *Identify the benefits but also the drawbacks and limitations of hierarchic energy coordination systems.*

While the first two research objectives focus on the coordination and its fundamentals in the context of energy and hierarchical composition, the third research objective focuses on the reflection asking questions like: What do hierarchical systems achieve? Does a hierarchical architecture reduce complexity? Does it facilitate the integration of EMS? Do hierarchical approaches support centralised or decentralised approaches? Which systemic effects can be stated? Addressing the benefits we should also investigate the limitations to find out what do hierarchic architectures not achieve and what problems arise.

In this thesis we provide the following contributions to address the research objectives from above.

- **Analysis of SES models and their usage in hierarchical structures:** We present models that are relevant for the coordination in SES and most importantly we show which information of those models is necessary for EMS interfaces and how they can be aggregated (composed) and disaggregated (decomposed) inside hierarchical coordination systems.

- **A generic hierarchic architecture for the coordination of EMS:** We suggest a generic hierarchic architecture that is based on best practises with sound architectural design patterns. The architecture specifically addresses hierarchical systems including their interfaces and behaviour that allows aggregation and disaggregation.
- **A framework to implement and test hierarchic SES architectures:** We develop and provide a technical framework as open source to evaluate hierarchic SES coordination systems based on co-simulation. The framework supports the implementation of different scenarios and the interconnection with industrial systems using state of the art protocols.
- **Two case studies to evaluate different coordination systems:** To evaluate our concept, we present two complementary case studies. The first case study investigates a scenario, where EMS offer flexibilities of power production units. The second case study investigates the coordination of building-EMS. For both case studies we use our generic architecture, our framework, design the interface information models based on SES models and use them to generate industrially usable interfaces.

Analysis of SES models and their usage in hierarchical structures

Our power supply system operation and planning relies on sound mathematical models. They are widely used throughout different systems: inside control applications; inside scheduling and optimization systems; as a foundation for energy markets that enforce an economical operation; integrated in design tools to layout the infrastructure, to configure and size the installed equipment; and of course to calculate costs, income or determine expected emissions. Depending on the purpose, such models vary in their granularity and associated details. The details depend on great extent on the systems at hand and on the time period that is of interest.

In this thesis, we focus on systems for (hierarchical) EMS coordination and on architectures to build such systems. Coordination belongs to models for optimized scheduling and partly covers control approaches. To design a sound generic architecture, we introduce a selection of related mathematical models. We analyse the models top-down in Chapter 3. We start with high-level coordination approaches based on market models. They improve the economical operation. We continue with more granular, physical models to present additional effects that effect the operation. Those models particularly address the distribution of energy, hence, the operation of energy networks. Finally, models for an optimized scheduling are introduced. These models address particularly the operation of individual components, such as generation and storage units that are controlled by an EMS. For each model, we describe the intention behind the model. The particular result of this analysis is the derivation of transferable information models that are used for the interfaces to establish an information exchange. Further, we demonstrate how the information models are used in hierarchical systems that aggregate and disaggregate their models from the specified interface data in Chapter 5 and Chapter 6.

A generic hierarchic architecture for the coordination of EMS

For the realization of SES with hierarchical systems, such as cells and their composition to new cells, work on the architecture is as important as work on power systems models, technologies and products. The architecture needs to define clear system boundaries in form of system interfaces, clear system specifications, so that the system engineer knows what functionalities must be implemented and provide a clear understanding about the composition of the related systems. To define an architecture that is as much generalizable as possible, we present a constructive derivation in Chapter 4. First, we describe the envisioned scope and typical requirements of the envisioned systems. After that, we derive the architecture using best practices from software engineering using well-known architectural design patterns. We explain them with their rational and show how they help us to achieve the construction of hierarchical systems.

After the architectural design patterns are clear, we describe the behaviour of single elements within the hierarchical system. Those elements are sometimes denoted as cells, prosumers, microgrids, but for us they are simply EMS. We describe the role of those systems and the most important behaviour in terms of interactions with their surrounding, i.e. how one specific EMS has to interact with other EMS that belong either to the category *parent* or *child*. The parent EMS is understood as a coordinating system, which process the data from the interfaces of its children. Based on that it derives an answer signal to coordinate its children. A child EMS is understood as a system, which realizes available coordination signals into actions. The result of this part is a better understanding about the involved system boundaries, necessary architectural design pattern to implement such systems and how the coordination is achieved in terms of a system behaviour.

A framework to implement and test hierarchic SES architectures

We provide a technical framework to test and evaluate the generic architecture. The framework is implemented as a co-simulation. It supports various hierarchic compositions with different information models and facilitates the implementation of SES scenarios using standardized common information models of power systems. It also allows to connect external systems, such as external EMS or external simulations. This opens the possibility for hardware in the loop testing, while the framework provides an environment for larger SES simulations. The framework is available as open source[3].

Two case studies to evaluate different coordination systems

We provide two case studies to demonstrate how the hierarchical architecture can be applied for SES. Both case studies are real prototypes that we implemented according the generic architecture using the developed co-simulation framework. The first case study investigates a control scenario with flexibilities of power producers that are interconnected with a virtual power plant. In this case study, the virtual power plant coordinates groups of power producers. Each group consist of a number of power producers. Depending on the type of production, each producer can adapt its production. The possible adaptation is a flexibility. It is offered to the higher component for coordination. We found that generic architecture is feasible. A hierarchy can be established.

[3] `https://github.com/SES-fortiss/SmartGridCoSimulation`, last accessed in March 2020.

Flexibilities can be aggregated and disaggregated. Industrially usable interfaces can be generated from the information model. The main difficulty is the choice of the right abstraction to model the data exchange, but power systems models help here significantly.

The second case study handles a more complex scenario with multiple buildings that are interconnected over an electrical network and district heating. We show their coordination when all buildings share their production capacities, storage systems and expected demand in a coordinated quarter scenario. Due to the slower reaction of the heat network, we explore here a planning problem. The results are the information model for the interfaces and an analysis that shows significant system benefits in terms of potential cost and CO_2 reduction of the energy supply. Further, we generate other technical interfaces here, namely OPC UA, which are widely used in the automation domain. We found also that system boundaries from mathematical optimization differ to those of the system architecture. Finally, we could confirm that our generic architecture, our framework, the design of information model for the interfaces based on presented SES models are beneficial to design hierarchic SES for both system planning and integration of real EMS.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces the related work. It gives an overview over the terms and related concepts, introduces work on other SES architectures, as well as methods and tools that are used in this domain for specification and analysis. Chapter 3 continues with the foundation of this thesis introducing different models for coordination of energy systems and describing how those models affect the design of hierarchic architectures. Chapter 4 introduces the generic architecture and presents the technical framework to test the architecture. Chapter 5 and Chapter 6 describe and evaluate two cases studies answering the research questions related to our research objectives. Finally, Chapter 7 summarizes the thesis and provides directions for future research. Additional material is contained in Appendix A and B.

Previously Published Material

Parts of the contributions presented in this thesis have been published in:

(Bytschkow et al., 2015) Bytschkow, D., Zellner, M., and Duchon, M.: *Combining SCADA, CIM, GridLab-D and AKKA for smart grid co-simulation.* In Innovative Smart Grid Technologies Conference (ISGT), IEEE Power & Energy Society, 2015.

(Bytschkow, 2016) Bytschkow, D.: *Towards composition principles and fractal architectures in the context of smart grids.* it-Information Technology, 58(1), 3-14, 2016.

(Bytschkow et al., 2019) Bytschkow, D., Capone, A., Mayer, J., Kramer, M. and Licklederer, T.: *An OPC UA based energy management platform for multi-energy prosumers in districts.* In Innovative Smart Grid Technologies Europe (ISGT Europe), IEEE Power & Energy Society, 2019

(Koß et al., 2012) Koß, D., Bytschkow, D., Gupta, P. K., Schätz, B., Sellmayr, F., and Bauereiß, S.: *Establishing a smart grid node architecture and demonstrator in an office environment using the SOA approach.* In Proceedings of the International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG@ICSE), IEEE, 2012.

(Duchon et al., 2014) Duchon, M., Gupta, P. K., Koss, D., Bytschkow, D., Schätz, B., and Wilzbach, S.: *Advancement of a sensor aided smart grid node architecture.* In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, 2014.

(Heidemann et al., 2019) Heidemann, L., Bytschkow, D., Capone, A., Licklederer, T. and Kramer, M.: *Sector Coupling with Optimization: A comparison between single buildings and combined quarters.* In 8th DACH+ Conference on Energy Informatics, 2019.

Related work that contributes indirectly to this thesis has been published in:

(Hackenberg et al., 2012) Hackenberg, G., Irlbeck, M., Koutsoumpas, V., and Bytschkow, D.: *Applying formal software engineering techniques to smart grids.* In Proceedings of the International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG@ICSE), IEEE, 2012.

(Irlbeck et al., 2013) Irlbeck, M., Bytschkow, D., Hackenberg, G., and Koutsoumpas, V.: T*owards a bottom-up development of reference architectures for smart energy systems.* In Proceedings of the International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG@ICSE), IEEE, 2013.

(Bytschkow et al., 2014) Bytschkow, D., Quilbeuf, J., Igna, G., and Ruess, H.: *Distributed MILS architectural approach for secure smart grids.* In International Workshop on Smart Grid Security. Springer, 2014.

(Bytschkow and Duchon, 2015) Bytschkow, D. and Duchon, M.: *Ladestrategien für E-Fahrzeuge: Koordination vs. Optimierung.* In ETG Fachbericht zur Fachtagung - Von Smart Grids zu Smart Markets. VDE Verlag GmbH, 2015.

(Bytschkow and Ascher, 2017) Bytschkow, D. and Ascher, D.: *CIM, Domänenmodellierung und Herausforderungen für Softwaresysteme im integrierten Verteilnetzbetrieb.* In Zukünftige Stromnetze für erneuerbare Energien (OTTI), 2017.

2 | Related work

In this chapter, we review the related work regarding the specification and analysis of SES systems. In particular, we discuss current activities that develop SES architectures, specifications that target system interfaces to improve system interoperability and approaches to evaluate SES systems in terms of architectures and new functionalities. In each section, we present the related work, show open issues in these fields and relate them to our work. For the interested reader, this chapter is a good entry point for further studies.

2.1 Architectures and Architectural Design Patterns for SES

Many countries, national and international organizations are interested in the development of SES. A particular interest lies in the specifications of the involved systems, because these have a large impact on the system vendors, power infrastructure, markets, customers and industries. The specifications are created from people with various backgrounds, from different sectors and numerous organizations with own goals and visions. These multi-disciplinary specification activities involve a high degree of interactions between people, software and hardware. Furthermore, different maturity levels of the involved systems, from sketchy SES concepts due to expected regulation policies to mature, industrial-strength solutions are considered. Hence, due to these clashes of interests, specifications are intrinsically complex. Sometimes, they are even seen as *wicked problems* (Irlbeck et al., 2013), a term coined by Rittel and Webber (1974).

Several approaches exist to reduce the complexity. Mostly, they involve abstractions to structure and describe the involved systems. We differentiate, between approaches that derive standards or domain specific architectures that we call reference architectures, generic architectures that use a predefined pattern to structure system functionalities and architectural design patterns that constrain the interactions of the involved components. We present those approaches and how they differ from our approach in the following.

2.1.1 Reference Architectures and Standards

Mature domains, such as communication, databases or automotive have well known reference architectures (Zimmermann, 1980; Sheth and Larson, 1990; AUTOSAR, 2018). They facilitate the development of the system, re-use of components and their integration. This lowers the costs and increases the quality.

SES do not have a worldwide valid reference architecture, due to the different national regulations within the power sector, which concern the responsibilities of the stakeholders, the sovereignty about the infrastructure, the related functionalities and access to data. However, there are several recognized references to guide the product developers and improve the interoperability. Two popular examples are the IEEE P2030 (IEEE, 2011) from the US and SGAM (ESO, 2012) from Europe. Both approaches tackle the challenge to define reference architectures for SES with a common terminology, different abstraction views, involved stakeholders and components, and most importantly standardised interfaces for the interaction.

IEEE P2030

The IEEE P2030 standard was developed to grasp the expected variety of the upcoming smart grid technologies in the US. It started with the NIST work on a roadmap for SES standardisations (NIST, 2010). The work categorised the expected systems to seven domains and related connections, shown in Figure 2.1. In a follow up step, the IEEE P2030 initiative continued the development to create a reference architecture.

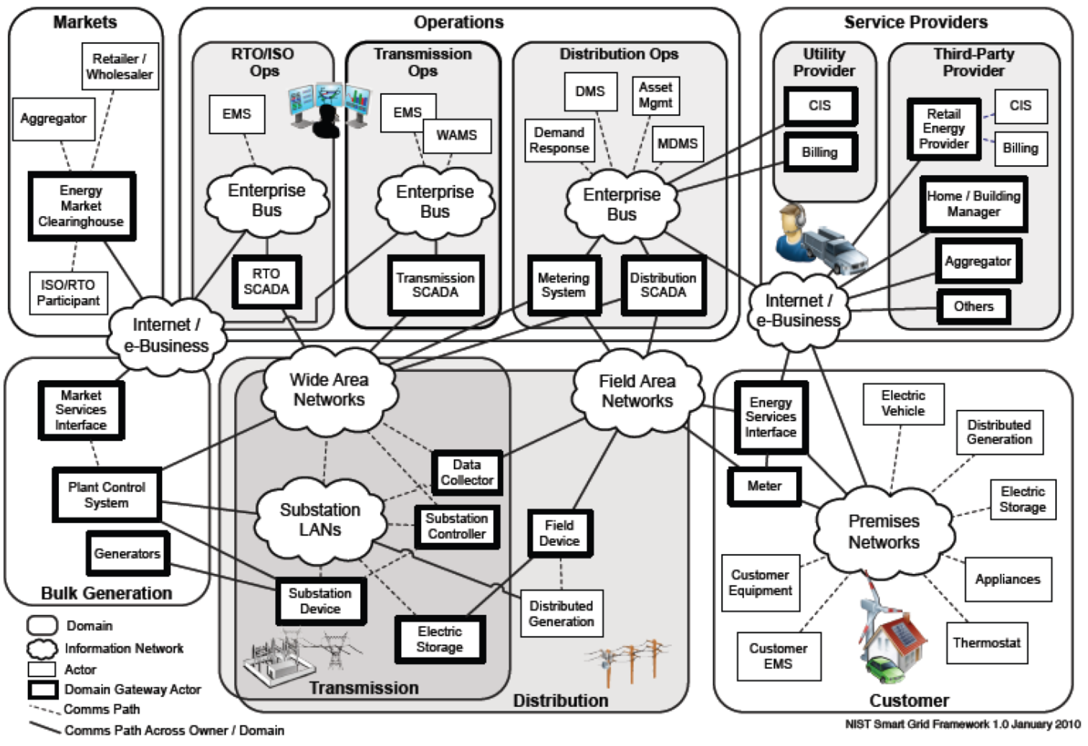


Figure 2.1: NIST Smart Grid Framework (NIST, 2010)

Inspired by the success of layered systems in communication and automation, IEEE P2030 introduces for layers for their reference architecture. It defines three layers on top of the presented domains from the NIST Smart Grid Framework, a power system layer (PS-IAP), a communication technology layer (CT-IAP) and an information technology layer (IT-IAP). To harmonize the interoperability, components are allocated to those layers (power system hardware to PS-IAP, gateways to CT-IAP and energy applications to IT-IAP) and their roles are shortly described in a comment. Further, their interaction partners (other components) from the same layer are identified. Each identified channel is labelled and shortly explained as well. However, this is done on a very informal and rough basis. For instance in CT-IAP, two components like the market and the public internet are connected over the channel CT26. This channel is specified as: "*It connects the market with utilities and other third-party providers through the public Internet.*" (IEEE, 2011). Other examples are similarly vague and links between the layers do not exist at all. Therefore, this approach is applicable for classification, but it does not help system developers to implement solutions. To address this issue, a European initiative has adapted the presented approach in their EU Mandate M/490, the Smart Grid Reference Architecture Model (SGAM).

Smart Grid Reference Architecture Model (SGAM)

SGAM is a European approach to develop a reference architecture for smart energy systems (ESO, 2012). It uses initially a similar systematic approach as NIST and the IEEE P2030, but extends it with more fine granular layers and a complementary perspective that represents the hierarchical levels of power system management, which they call zones (Figure 2.2). This supports the idea of aggregation. Instead of a predefined top-down approach of components, SGAM provides a more generic framework to work with use cases and structure them in the reference model.

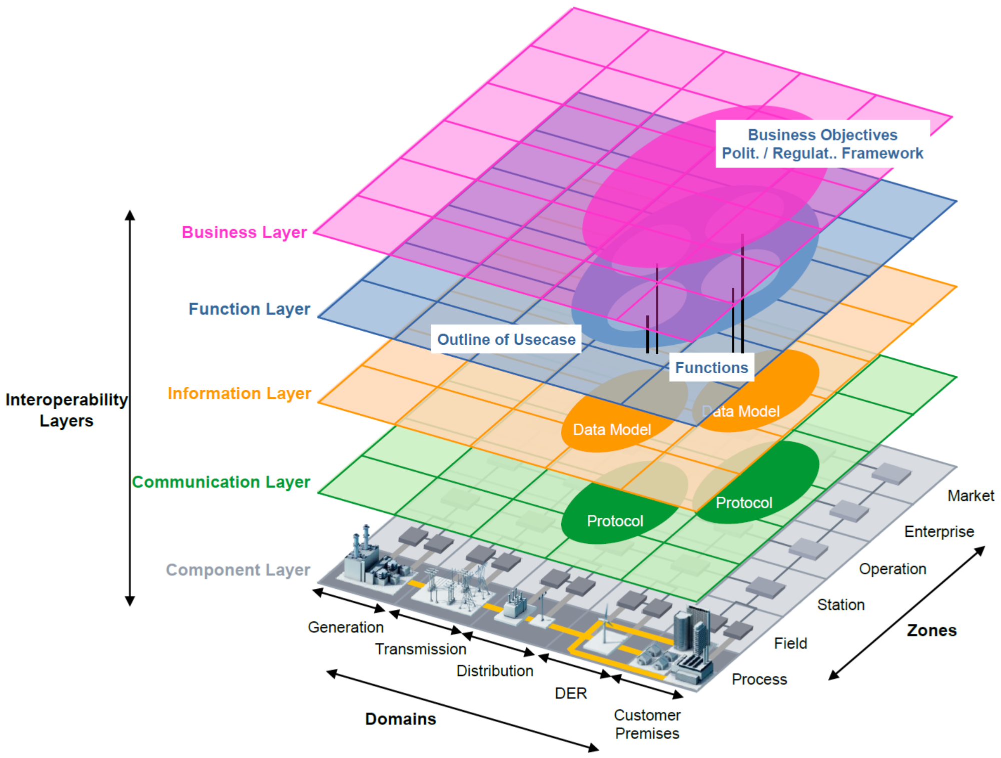


Figure 2.2: SGAM reference framework (ESO, 2012).

The idea behind the framework is to develop SES use cases using a model based approach and map the model elements that are of interest for the specification to the framework's structure. This has two benefits. First, the approach is more tailored for development using classical software requirement elicitation techniques like use cases. This allows to develop more granular specifications of the intended systems and map them into a general framework. Second, the mapping allows to collect use cases, store them in a database and provide an indexed library, where other developers can search and extend the available use cases and related standards.

The application of such approaches have been demonstrated, for example by Dänekas et al. (2014). In addition, modelling tools have been developed, for instance the SGAM toolbox that is described by Neureiter (2017) to support the development with a more structured process. Although the approach looks promising at first glance, the difficulty is hidden in the details. SGAM offers several zones for hierarchical systems. It envisions that systems of different zones have different responsibilities with different data models and different protocols. However, the main problem of integrating complex EMS is not related to finding the right zone, domain and layer. It is more pragmatic. It focuses more on the interface and behaviour specification of SES components, and for coordination they need to correspond to the underlying mathematical models. So it is similar to cross-cutting issues, as those that have been addressed by the cross-cutting SGAM teams for system aspects and data communication interfaces (see Section 5.1.3 and Section 5.1.4 in ESO (2012)). The cross-cutting issue teams particularly emphasise amongst others the importance of semantics and formalisations of functional requirements at application levels, as well the harmonization with data models such as the common information model (CIM), which is an independent standard for the specification of data at the interfaces in the energy domain.

Common Information Model (CIM)

CIM represents the state of the art standardisation efforts in the energy domain. A good overview about CIM is given in the book by Uslar et al. (2012) or the official introduction (CIM, 2015). More recent activities are presented by the CIM user group[1]. The standardisation activities span three major standards: IEC 61970-301[2], IEC 61968-11[3] and IEC 62325-301[4] that cover different areas of the power system domain.

The authors work with a model-based approach for their specifications, mainly using the unified modelling language (UML) to provide a technological independent and abstract description of the data that exist in the power domain. With the strong focus on modelling power systems, particularly on electrical equipment, distribution management and markets, CIM has a clear focus on the interoperability and data exchange of current centralized applications within the energy domain. Architectural questions are not in the scope of CIM. It therefore lacks in capabilities to specify architectures, or to describe the behaviour of the intended systems. But it is still very useful as an internal model within applications to represent power systems.

Bottom-up Development of Reference Architectures

Reference architectures offer a good starting point for the development process and provide orientation. But unfortunately, the standardization bodies IEEE and European Standardizations Organizations (ESOs) specifically demand to use top-down approaches (CIM, 2015) following their classification, while the development of innovative solutions is often organized bottom-up, where researchers or companies build prototypes to try new approaches or system architectures and test their application in field tests. Therefore, collecting the information gathered from those solutions is another important contribution for specification activities. To approach these development efforts Irlbeck et al. (2013) presents a complementary bottom-up approach that is based on an incremental extension. In his work, the author develops a method to develop SES reference architectures based on architectures of systems with different maturity. The intention is to enable that the reference model can be used and extended by non-experts, but still provide

[1]The CIM user group: `https://cimug.ucaiug.org/`, last accessed in March 2020

[2]IEC 61970 - Part 301 is the base model of power system components and their relations at the electrical level.

[3]IEC 61968 - Part 11 is an extension for the asset management, work scheduling and customer billing.

[4]IEC 62352 - Part 301 is another extension that covers the required data exchanges for electricity markets.

enough formality to analyse system properties or identify important interfaces for standardization. This approach led to a broader reference architecture (Irlbeck and Koutsoumpas, 2015), which is particularly useful to aggregate developed solutions and provide a bottom-up specification model for novel domains.

Relation to our work. The line of research that we presented above is the state of the art to reduce the complexity of SES and facilitate the development of accepted standards and clearly defined interfaces for interoperability. The presented approaches help to classify systems and structure use cases according the defined layers, domains or zones. The classification allows to compare new applications or systems with systems in the same field, or with systems that are in the nearby.

In our thesis, we consider the presented reference solutions, but we focus only on one specific part of it, namely those applications that fall into the category of a hierarchical coordination with EMS that offer similar functionality and interfaces, including virtual power plants, prosumer and microgrids as important flexibility providers. We present them in Chapter 5 and Chapter 6. Our topic is therefore less broad than the presented approaches, since we do not want to give a reference architecture. Instead, we present a method and specification support to for hierarchical coordination architectures and present their application in the energy domain. Nevertheless, we use several components, that can also be found in IEEE P2030, SGAM or the reference architecture in Irlbeck et al. (2013) and Irlbeck and Koutsoumpas (2015). In addition, we use the data models of CIM (2015) for our technical implementation (see Section 4.3) as a standardised representation of the power network and to provide a technology for evaluation that is as close as possible with real industrial applications.

2.1.2 Generic Architectures for SES

Another line of architecture research for SES tries to reduce the complexity by finding commonalities for the operation and control of power equipment. The idea is to offer power resources (production and consumption) to other SES components, to reach a better balance between the supply and demand of power. That balance is of paramount importance for power supply networks to achieve a stable operation. Architectures for systems that focus on these commonalities are often designed with a generic approach in mind. Different generic approaches exist, but the core idea is to create a cooperation of similarly designed interactive systems. In the following, we present the work on generic architectures for SES from several research groups.

Prosumer-based Architectures

The research from Moslehi and Kumar (2010) analyses reliability challenges in the power network due to increasing renewable energy sources. They find that most of them can be solved with available technologies. Particularly, load management and demand response technologies are promising. To consider the distribution and large amount of components, they suggest a flat architecture, where each system offers some particular functionality that can be consumed by other systems. Instead of central SCADA systems, they suggest to use “intelligent functional agents” for temporally coordinated hierarchical monitoring and control actions.

Another work is presented by Grijalva and Tariq (2011), who goes in the same direction and proposes a prosumer-based smart grid architecture (Figure 2.3 left hand side), to enable a flat, sustainable power supply. The prosumer is the basic entity in his work. He requires that a prosumer internally has a layered architecture as in automation systems and specific applications to interact with other prosumers directly. He argues that the smart grid can be designed as a network of such prosumer systems. At the same time he shows an architecture of such an autonomous generic prosumer (Figure 2.3 right hand side) that contains several applications.

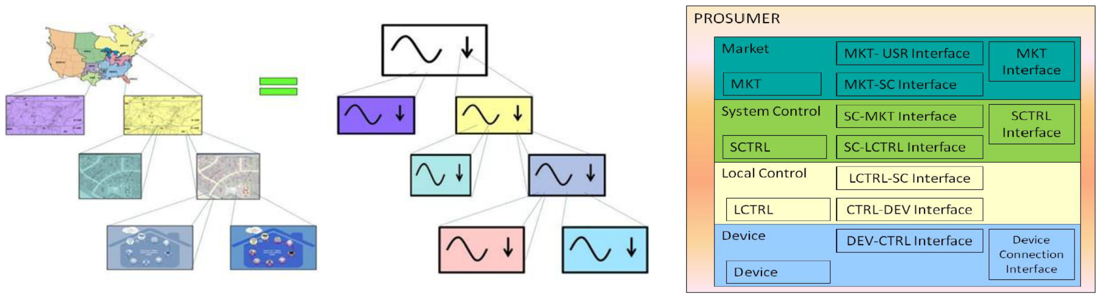


Figure 2.3: The prosumer oriented system architecture as depicted in (Grijalva and Tariq, 2011)

Even though the ideas are clearly formulated, the authors stay very generic. They also give only little details about the intended interfaces stating only very rough ideas. For instance, they say for the *local control service (LCRRL) interface* provides operations called “Capability”. But they omit to explain how such capabilities are specified and how they are used in the system. They also do not present whether every prosumer has the same set of applications and whether every prosumer exposes all of its interfaces at each layer to external components or not, and if there are particular roles for prosumers, like utility prosumers or home prosumers and how they differ. Without further refinement and a more concrete specification, system developers cannot realize such systems since many details remain unclear or even ambiguous.

Service Oriented Architectures

The service oriented architecture (SOA) approach (Papazoglou, 2003) is another generic approach to create an architecture for SES. In SOA the complexity is handled by encapsulating parts of the required functionality as services that are well-defined, self-contained software components, which provide standard programming interfaces and are independent of the state or context of other services. A published service interface provides a common interface, through which any application can be accessed independent of its concrete implementation (Papazoglou and Van Den Heuvel, 2007).

It has been demonstrated that EMS systems, e.g. prosumer (Koß et al., 2012; Duchon et al., 2014), or demand response applications (Chrysoulas and Fasli, 2017) can be created using SOA. In a SOA each software component is designed as an individual service, which offers a defined interface to other services. To receive data it subscribes to other services. The communication is realized either over common internet based protocols, such as TCP/IP or over dedicated communication buses. This means that every service sends and receives messages that are defined by the interface over dedicated communication implementations. Shared knowledge between different services is avoided. Two goals are achieved in this context: A generic approach for system interaction in a heterogeneous hardware landscape, which offers generalized interfaces for data, and higher-level services for data-analysis, event processing and control applications.

An overarching architecture for SES can use SOA concepts, for instance to create a system that is similar as envisioned by Grijalva and Tariq (2011). Nevertheless, the freedom to create any service with SOA is just an enabling technology. Without additional restrictions or requirements to follow certain SES standards, there is no guarantee that SOA based SES systems interacts in a well defined way. Therefore, the success of a SOA architecture is directly coupled to additional restrictions and specifications for the desired behaviour and data that is exchanged over the interfaces.

Holonic Architectures

Holonic architectures are another way to construct hierarchic, flexible and agile systems. A holonic architecture is an architecture for a system that consists of interacting holons. A holon is an interactive software component, for which autonomy and cooperation are the two essential features. It was introduced by Koestler et al. (1967).

Several approaches exist that uses holons to design generic SES architectures, e.g. the work by Frey (2013); Frey et al. (2013) and by Negeri et al. (2013). A holon is the basic entity in both approaches. It represents a prosumer system, or more concrete the external visible "gateway" component, as well as smaller autonomic devices and also systems that integrate and manage multiple prosumer systems. Frey (2013) develops in his work the generic architecture from the context of Autonomic Computing with its *monitor-analyse-plan-execute* (MAPE) cycle (shown in Figure 2.4), which was introduced in the work of Kephart and Chess (2003).

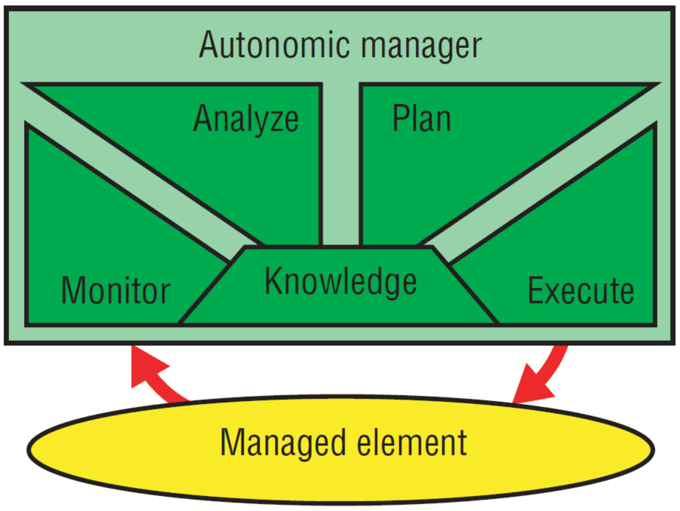


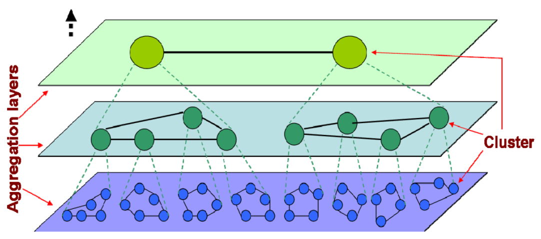
Figure 2.4: The MAPE control architecture as introduced by (Kephart and Chess, 2003).

In this cycle, there are labelled software components that implement several dedicated functionalities. The first one is denoted as *Monitor*. It represents the component that is responsible to collect the required data. The next component that is denoted as *Analyze* covers data processing.

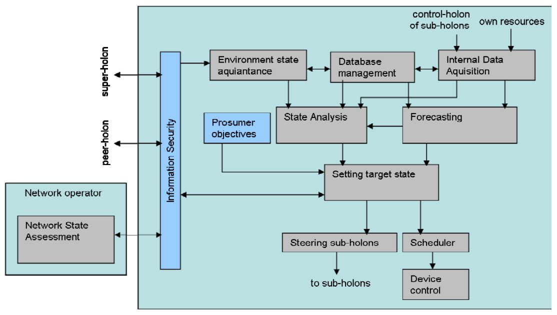
It calculates the current system state or identifies states that requires follow up actions. The next component called *Plan* is responsible to plan future task executions. Finally, the execution of certain actions that lead to a new system state is denoted as *Execute* component(s). If the components share common knowledge, e.g. a common database, the architecture is often denoted as MAPE-K architecture. The MAPE-K cycle forms a control loop around the *managed element*, which represents the system of interest, for instance a physical system or a server infrastructure that runs multiple software components.

Following the Autonomic Computing vision, Frey (2013) designs holons as components with SES specific functionality. The holons cover the different areas of monitoring, analysis, planning and execution as a MAPE based architecture. Frey introduces different applications for a house and a district scenario, where he structures different holons, similarly to different SOA services as sub-components of the house or the district EMS. The holons are autonomous software components that interact via message exchanges. His work focuses on the question how the system handles multiple-objective goals with its generic architecture.

Another comprehensive work with holons as the main architectural element is presented by Negeri et al. (2013). He uses holons to create an architecture for a control system, where holons provide several control services and are arranged in a holonic hierarchy (Figure 2.5a). The services are available using the SOA approach (Figure 2.5b). Negeri describes in his work the services of a holon, derives the control architecture and presents the results for a coordinated activity to achieve a peak reduction in a network of houses with photovoltaic systems, electrical consumption, electrical vehicles, and micro-CHPs.



(a) Organisation of holons.



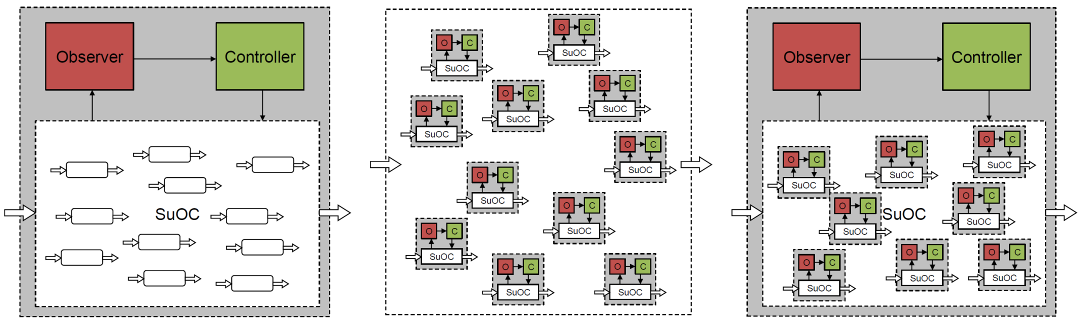
(b) Architecture of a control holon.

Figure 2.5: Holonic SES systems as depicted in (Negeri et al., 2013).

The holonic concept has a clear focus on autonomy of software components and their interaction over communication. Every holon has a clear system boundary - its interface - over which it receives or sends messages. In addition, holons are part of larger systems. This hierarchical structure is sometimes denoted as a holarchie. Beside these restrictions, the holonic concept is rather open to create any kind of system with any kind of different holons inside this system. This is also reflected in both approaches, either from Frey (2013) or Negeri et al. (2013). Both approaches introduce different types of holons inside their systems, which cover different functionalities, have different interfaces with different messages for their communication. The generic aspect in Frey's architecture is present through the convention that every component implements a MAPE cycle and a way to approach a multiple goal environment. There is no enforcement of a similar interface or a defined behaviour including certain criteria for aggregations or disaggregations for each EMS in that approach. In consequence, even if you argue that each EMS has those particular functionalities like monitoring, analysing, planning and execution, there are no particular restrictions that enforces each EMS to offer similar data at their interface. In consequence it might happen, that each system is again individual with respect to its interface and behaviour, even if internally holons are denoted with equal labels. Negeri's work describes that each holon offers a set of services with interfaces according the SOA principles. All services are presented as internal parts of control holons. From the behaviour point of view, the authors describe their major use case as a negotiation process that relies on different behaviours of holons that belong to different organisational layers. Therefore, it remains unclear if the holons from different organisational hierarchies are generic with respect to their interface and behaviour or if they are again individual systems with individual interfaces. Thus, the holonic concept is promising, but it requires further work to create a generic architecture for a system with suitable, interoperable interfaces and the desired hierarchic structure.

Organic Computing

Another approach that aims to reduce complexity for the control of large distributed systems, such as a smart grid, is *Organic Computing* (Schmeck, 2005). The name "organic" reflects its inspiration from biology, where systems are self-contained entities and interact with each other to achieve some common goal. Organic computing is similar to the MAPE-architecture and introduces common terms to improve the structure and with that reduce complexity. The generic architecture of the approach was introduced by Richter et. al (2006) as an *observer/ controller architecture* (OCA). The observer collects and aggregates information about the system under observation and control. The controller receives the aggregated values and takes appropriate action to influence the system. The focus on these two components particularly emphasizes the fact that input / output specifications are essential to define proper software architectures. The OCA can be customized to meet the requirements of different scenarios. At least three main architectural options are available: (a) central, (b) decentralized / distributed, and (c) multi-level / regional / hierarchical (Schmeck et. al., 2010). They are shown in Figure 2.6.



(a) Central (b) Distributed (c) Multi-Level

Figure 2.6: Possible realisations of the generic Observer/Controller architecture as depicted in (Tomforde et al., 2011).

For smart energy systems, a multi-level hierarchic OCA is presented in the context of smart homes. An OCA is used to create a flexible middleware, with the objective to integrate different energy specific applications (Allerding, 2014). A local observer-controller [O/C] component is assigned to each smart home device. Local O/C-components implement the required device functionality. A global O/C-component receives the filtered and aggregated information of local O/C-components to obtain a global state of the system. Based on that it calculates forecasts and uses optimization techniques to calculate the desired behaviour of the whole system (Becker, 2014). The consideration of multiple energy networks (sector coupling) with OCA is presented by Mauser (2017). Note, that internally, O/C components can be modularly designed using layers and exchangeable O/C-units, such as different optimizers or hardware abstraction components as described by Mauser (2017). The OCA work focused on the development of a particular building energy management system representing a prosumer. Therefore, even if OCA might be promising, beside structure, the application of the generic hierarchy with common interfaces as an underlying architectural concept is still an open question for OCA as well.

Cellular Architectures

Finally, there is another recent concept that is currently widely discussed in Germany in multiple research projects. It envisions the definitions and investigations of new generic architectures that enable nested hierarchic EMS structures with different levels. It is denoted as a cellular architecture. Here, the most popular and recent work is based on activities that are triggered by the VDE study written by Benz et al. (2015). In their work, the authors analyse the potential for energy coordination in local area networks and demonstrate how it helps to improve the network's operation. The work does not introduce holons, agents or any other form of interactive or *intelligent* software systems, it focuses rather on the power technology, current consumers and achievable energy balances using flexible shifting capabilities similar as Moslehi and Kumar (2010). The authors show that the system can benefit from a generic architecture approach, introducing an analogy with cooperating cells (also inspired from biology), where a cell (an area) is formed from available flexible power components, and smaller cells form larger cells. However, a concrete definition for a cell and even simple details, like the involved control systems with the corresponding system boundaries including the data input/output specifications, are missing.

Relation to our work. The presented work of the authors for generic architectures pursue similar ideas and think in the same direction, as we do in our work. Especially Moslehi and Kumar (2010); Grijalva and Tariq (2011); Benz et al. (2015) present the potential of the idea very nicely. Nevertheless, the authors remain rather informal in their studies and omit many important details for systems developers. The work of Allerding (2014); Becker (2014); Mauser (2017) and Frey (2013); Negeri et al. (2013) provide more concrete examples. They present several technological case studies. However, it seems that the ideas in those examples to establish generic architectures are mainly driven by structural concepts like the MAPE cycle or OCA and complemented by the emphasis on autonomic nature in the holonic approach. However, it is often not clear if the presented approaches provide similar interfaces and behaviour, as expected for instance by Moslehi and Kumar (2010). Several design and implementation questions remain therefore open.

In our work, we also approach the desired hierarchic system over several layers. We use the ideas from the presented examples, for instance the fact, that each software system, be it a holon, or a prosumer, is in fact an EMS with an interface that provides the details for available resources. We give a clear specification about what an EMS is, describe its interface and the expected behaviour of such a system to utilize the resources. Additionally, we extend and strengthen the previous work with architectural design patterns to improve the used software engineering practises. We demonstrate the usage of well-known interface technologies and concepts such as REST and OPC UA to show the applicability in a more industrial context. Internally, in our technical implementation of the behaviour logic we use similar software technologies as Frey, for instance the akka[5] framework. We refer to Chapter 4, where we introduce those concepts and developed technologies.

2.1.3 Architecture Styles and Architecture Design Patterns

In the previous section, we present general architectures that are used to design novel smart energy systems. However, there exist another complementary perspective from the software engineering domain that greatly facilitates the development of systems, which is often denoted as architecture styles or architecture patterns. It captures best practises from software engineers that have solved reoccurring problems multiple times and defined specific styles and patterns to approach architectural questions. It particularly allows dividing important functionality of component and structuring them in clear and efficient manner. In the following, we shortly introduce this perspective.

The terms architecture style and architecture pattern are both used to categorise architectures by describing their commonalities and involved design decisions. The usage of the terms is based on different definitions and interpretations, and is therefore often inconsistent and sometimes confusing. We do not want to provide an in-depth discussion and elaboration about all the different nuances and possible interpretations of this topic, but rather point to existing literature that deals with this issue and gives a pragmatic interpretation that is used in this thesis.

A detailed discussion about architecture styles, architecture patterns and their differences can be found in the work of Taylor, Medvidovic and Dashofy (Taylor et al., 2009). After defining a software architecture as a set of principal design decisions made about the system, they differentiate between an architectural style and an architectural pattern providing the distinction that a style is a collection of design decisions, which are context oriented, and a pattern is a collection of more specific, problem oriented design decisions. Further, the term style is used to designate strategic decisions and pattern for tactical decisions. Therefore a style differs from a pattern in its scope, abstraction and the relationship between the design decisions (Taylor et al., 2009). A clear differentiation relies, however, on an interpretation, whether a design decision is abstract enough to belong to a style or a pattern. Consequently, they admit that it is not always possible to identify a crisp boundary between both terms.

Another detailed terminology discussion is given by Fielding (2000). In his work, he elaborates the terms, architecture, its meta-model (i.e. the elements of an architecture) and architectural style, which he defines as a coordinated set of architectural constraints, that restricts

[5] `https://akka.io/`, last accessed in March 2020

the roles of architectural elements and the allowed relationships among those elements within any architecture that confirms to that style. He uses the term architectural style as a mechanism for categorizing architectures and for defining their common characteristics. He also compares the term with other definitions, in particular with the work by Perry and Wolf (1992), and the definitions by Garlan and Shaw (1993); Garlan et al. (1995) and Shaw and Clements (1997).

In contrast to a style, the term pattern does not necessarily target architectures. Instead, patterns often address implementation specific concerns, i.e. as a recipe for implementing a desired set of interactions among objects. For instance, SOA is rather a pattern than an architecture style following this argumentation. According Fielding, patterns are a well-proven possibility to solve a specific functionality on the implementation level rather than a set of constraints that restricts the instantiation of the architecture meta-model.

In fact, the definitions of the terms architecture style and pattern are often differently used, because they directly depend on the definition of the term architecture itself. And many software architects, including Taylor, Fielding, Perry, Wolf, Garlan, Shawn and Clements, have an own definition what an architecture is. However, two common observations can be made: (i) Styles and patterns are used to classify architectures and obtain some kind of architecture classes, and (ii) a pattern is usually closer to implementation than a style. To avoid further confusions between those terms, we use the term architecture design pattern (see also Marmsoler (2018)) in the following.

Several widely accepted architecture design patterns have large impact on system development. We just name a few to give concrete examples for architecture design patterns. For further explanations and more detailed examination of this topic we refer to Taylor et al. (2009). A very successful (and simple) architecture design pattern is the so-called *client-server* model. This architecture design pattern knows two elements, the client and the server. The client always initiates the communication. It sends a request to a server, which takes the request, performs the required actions and replies to the client. The constraint is that two clients cannot interact. Another successful example is the *layered architecture* design pattern, which is the underlying concept for current communication technologies. Further examples for architecture design patterns are *peer-to-peer, publish-subscribe, pipe-and-filter* or the *blackboard* style.

Working with architecture design patterns significantly helps to design systems, since it enforces to define concrete roles of components and how they interact. At the same time many libraries and technologies support the implementation of such systems. Today, we observe many SES systems that work with architectural design patterns, including SCADA systems with layers as well as client-servers (Daneels and Salter, 1999; Stouffer et al., 2006), SOA systems where layers, publish-subscribe and client-servers occur (Duchon et al., 2014), the presented OCA and holonic systems that have layers for hardware and service abstractions (Frey, 2013; Allerding, 2014), but also peer-to-peer architectures that are discussed in vision papers (Lasseter, 2011) and implemented for specific applications for market scenarios in combination with blockchains (Thut, 2018). But often those architecture design patterns are not explicitly addressed and the purpose for those decisions is not clear in the SES context. This leaves a lot of interpretation room for systems developers and leads to an inconsistent understanding of the intended system complicating further progress on system development.

Relation to our work. In this work, we do not want to conduct a deep discussion on the terminology and differentiate between an architecture style and a pattern. Hence, we prefer to choose only the term *architecture design pattern* for our categorisation and use constraints that refer to architectural elements and their composition. Our definition of an architecture design pattern has two integral parts. The architecture meta-model defines the elements of an architecture and their composition. Constraints define the propositions that refer to those elements and their composition. Consequently, if a system architecture is an instance of the architecture meta-model and fulfils all of the constraints, than it is an admissible system architecture that adheres to the architecture design pattern. Figure 2.7 presents a schematic representation of our definition what an architecture design pattern is.

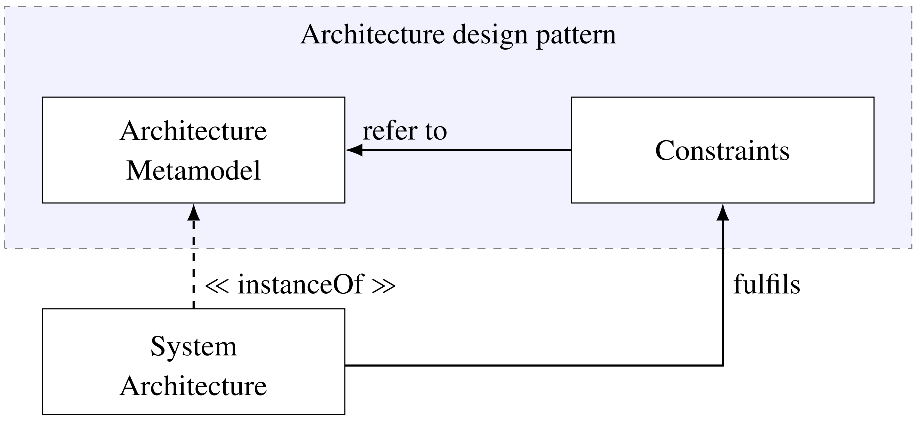


Figure 2.7: A schematic representation of an architecture design pattern and its relations to a metamodel, constraints and architectures.

A consequent application of architecture design patterns to real systems result in a set of advantages and limitations. One advantage is a greatly improved technical understanding of the system's structure and capabilities. Another one is the technical support to reuse and integrate components. A good example for this is the plethora of components that supports server and client architectures, or technologies to support the enforcement of security in a layered system. The good aspect about architecture design pattern is that they can be combined and as long as they do not interfere with each other. The combination yields basically a combined set of advantages and limitations (Fielding, 2000). We describe in Chapter 4 how we combine several basic architecture design patterns to establish our own hybrid architecture design pattern that helps to implement a generic architecture for SES systems.

It is worth noting here that all approaches, the design of reference architectures, for instance IEEE P2030, SGAM or Irlbeck et al. (2013), generic architectures, for instance Frey (2013) and Grijalva and Tariq (2011), as well as architecture styles presented by Fielding (2000) and Taylor et al. (2009) support the development of the SES vision, by providing a clear terminology and present possible solutions. These efforts facilitate the integration of components describing the roles for interconnection and the involved interfaces, but each with another abstraction perspective. This allows re-using specifications, code and technologies for the interfaces. It simplifies the integration efforts, reduces costs and increases the quality of the systems.

2.2 Technical Support for the Evaluation of SES

During the last years many different models, coordination algorithms and architecture concepts for SES have been developed and evaluated with respect to achievable improvements regarding costs and emissions or with respect to the technical feasibility. Therefore, different kind of simulations or demonstrators were developed. The conducted research activities can be distinguished into systems that represent energy nodes, such as buildings or microgrids and related EMS, and systems for coordination of power networks, which either address EMS for the operation of energy networks, e.g. SCADA systems, with dedicated network simulations or solutions that consist of multiple combined simulations, which we understand as co-simulations. In the following, we present both approaches and additional technical frameworks that support those activities.

2.2.1 Living Labs and Smart Buildings (EMS for nodes)

Recently, many new technologies have been developed and installed at the lowest level of our power supply system, including photovoltaics, storage, CHPs and electrical charging stations. The demand that those solutions are interactive and support our supply networks more actively leads to the development of local EMS. In the energy community, such systems are called prosumer, microgrids, living labs or simply smart buildings. The terms address similar research directions, where the idea is to create concrete systems and evaluate their potential impact with real demonstrators. Of course, each solution is often implemented differently. Nevertheless, the solutions discuss similar functionalities, similar technological stacks, similar architectural ideas and similar energetic flexibilities. Therefore, their development and the experience collected during those activities is extremely useful, when the experience is transferred into real applications.

There are plenty of demonstrators available together with very good overviews for that domain, e.g. by Becker (2014) or Mauser (2017). We therefore refer to their work, instead of repeating the plethora of details. In their work, Becker (2014) develops together with Allerding (2014) an EMS for a prosumer system based on the organic architecture approach, that we presented in the previous section. Their demonstrator is therefore sometimes denoted as the organic smart home. The system is integrated into the *FZI* [6] *House of Living Labs* (HoLL) and can be used to study the impact of buildings that are aware of their consumption, can adapt their consumption depending on the user's comfort and interact with its environment. A technical overview of the HoLL demonstrator is shown in Figure 2.8 and its software architecture is illustrated in Figure 2.9. Mauser (2017) extended the demonstrator to cover particularly multiple energy carriers and derived more advanced control algorithms for the EMS.

The development of the HoLL demonstrator towards a multi-energy system follows the general trend of combining more energy systems from various sources, as the combination of several available components promises a better energy utilization and with this another improvement towards less carbon emissions. This demands additional extensions of building's EMS making them more complex, but enabling also the usage of further flexibilities. Therefore,

[6] A research center related to the Karlsruhe Institute for Technology (KIT), Germany

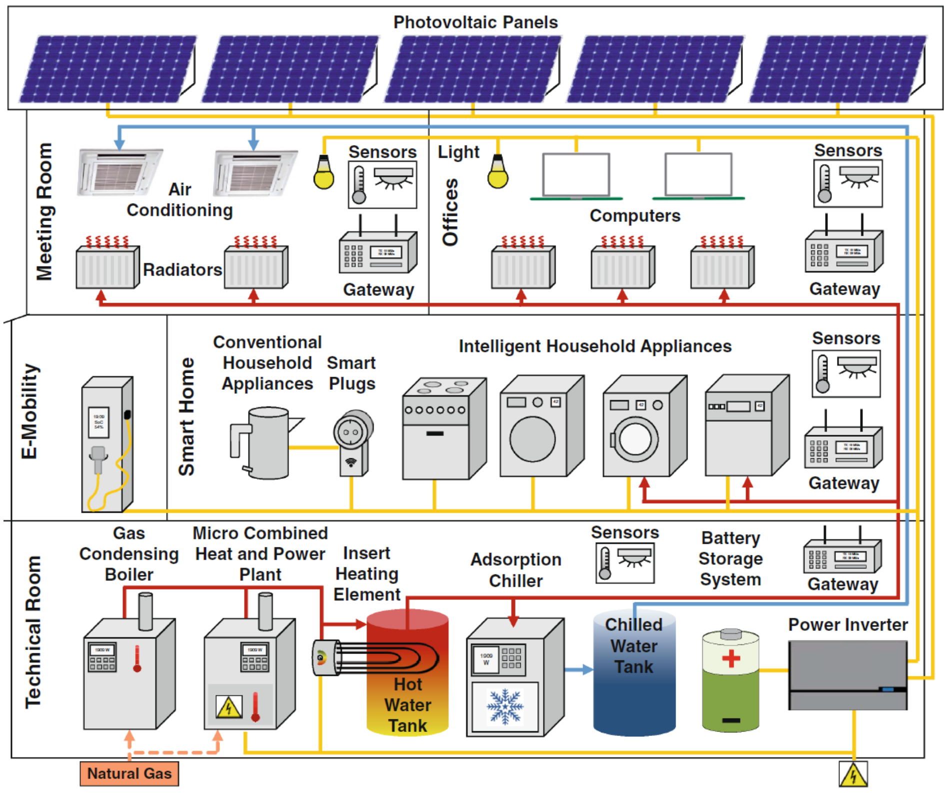


Figure 2.8: HoLL: Technical overview according (Becker et al., 2015)

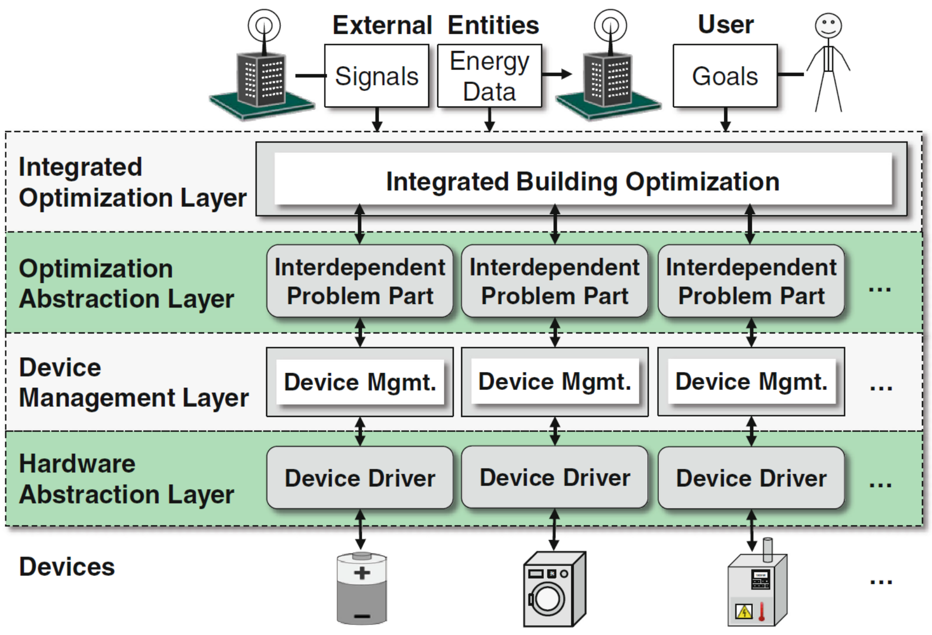

External Entities
Signals
Energy Data
User
Goals
Integrated Optimization Layer
Integrated Building Optimization
Optimization Abstraction Layer
Interdependent Problem Part
Interdependent Problem Part
Interdependent Problem Part
...
Device Management Layer
Device Mgmt.
Device Mgmt.
Device Mgmt.
...
Hardware Abstraction Layer
Device Driver
Device Driver
Device Driver
...
Devices
...

Figure 2.9: HoLL: Software architecture according (Becker et al., 2015)

more multi-energy solutions for Living Labs and larger systems such as quarter solutions and microgrids are being further developed. A good example for the next development stages is the Center for Combined Smart Energy Systems (COSES)[7]. The goal here is the experimental demonstration how individual buildings can interact with multiple energy networks and the validation of power systems models that are currently used for control or within simulations. The research activities include several perspectives, such as the supply of heat and electrical power, communication, control and energy management that provide available flexibilities of the buildings and a connection with external software systems including specific data and interfaces.

Relation to our work. At our research institute fortiss we followed a similar approach as (Becker, 2014; Mauser, 2017) and developed our own SES Living Lab demonstrator with several applications, such as predictions and demand response with interfaces in order to interact with external SES environments. The Living Lab is described in the work by Koß et al. (2012) and Duchon et al. (2014). It serves us as a prototypic environment to study new methods and control strategies for the operation of the building, study available protocols and to define interfaces for potential interactions with the environment, for instance how to offer flexibilities and how in implement them internally. It also helps to gather data, improve applications for instance energy forecasts (Rottondi et al., 2015) and try machine learning approaches (Bajpai, 2018). It was also used in multiple European research projects to integrate it with other Living Labs amongst other with the HoLL environment and as a technology to coordinate several hierarchic, self-similar energy nodes in India as presented by Gupta and Duchon (2018). The Living Lab serves as a reference to implement co-simulations such as virtual power plants (Chapter 5) or coordinating platforms (Chapter 6). Further, our system will be used at the COSES research centre to study the interaction of buildings and how to communicate optimized schedules to the building's EMS, without interfering too much with their internal measuring and control technology.

2.2.2 Co-simulations (EMS for Networks)

Another way to study the impact of interactive SES solutions and their effects on the network relies on the usage of simulations. They are a widely used approach to study and analyse the behaviour of systems, particularly when the systems are complex and many details exist. Simulations allow to represent novel control strategies and processes that appear in SES, such as load scheduling, market services or charging of electric vehicles, and execute them under various conditions. This way, the results and impacts created by novel SES solutions can be analysed.

Usually, simulations are designed to analyse particular parts of the system. In SES those are for instance physical power flows, energy costs and emissions in particular market designs or communication delays in real time control systems and their effect on stability. The simulations work with certain assumptions and detail abstractions. Not all details are covered in a single simulation. When novel systems with innovative control approaches or market interactions appear, researchers need to decide, which simulation is suited best to analyse their impact. Sometimes more than one simulation is required. Therefore, the idea for combined simulations

[7] `www.mse.tum.de/en/coses/`, last accessed in March 2020

(co-simulations) receives more and more attention. Industrial simulation tools seldom provide a full support for all kind of interactive systems with different inputs and protocols from their environment, because it is a great overhead for implementation. Therefore, co-simulations become more attractive. Their excellent use is demonstrated by several software researchers in the SES domain (Basso et al., 2013; Rohjans et al., 2013; Schütte, 2013; Molitor et al., 2014; Steinbrink et al., 2019).

Co-simulations are particularly useful to study novel applications and investigate their architecture design decisions to create real running systems. Our aim is to study several use cases that include different coordination approaches, like the determination of optimized schedules for power supply, demand and storage, but also analyse how flexibilities within power supply or demand can be represented so that are offered as services over interfaces for control applications. Therefore, two major requirements need to be covered in our approach: We need a very flexible modelling framework that supports the implementation of our expected services as part of dedicated software components or agents that communicate with each other. The communication topology and message sequences shall be as flexible as possible. Additionally, we need the simulation of the physical power network, including different physical properties, such as power flows, voltages as well as different levels and topologies. This is the second major requirement.

A number of related tools that meet the two major requirements have been developed during the recent years. One early example is the Epochs framework (Hopkinson et. al., 2006), which couples an agent based simulation with dedicated power system simulation tools using the high-level architecture (HLA) approach. An extension of that framework was introduced by Lin et. al (2011) in order to improve the accuracy due to the mixture of event based communication and cycle based calculation of the power system simulation.

Another approach is MOSAIK, that is available as OpenSource[8]. MOSAIK was first introduced by Schütte et al. (2011). It is designed as a co-simulation environment with an extended concept for coupling agent-based and power flow simulations, in particular, for large-scale simulations, including generation and parametrization of models. Many other tools are available that go into the same direction. A good overview is given for instance by (Schütte, 2013) that presents the different approaches and compares their benefits. Another broad summary about tools that are used in that domain is also provided by Mauser (2017).

Relation to our work. Due to the context of our live running Living Lab demonstrator that exist at our research institute, as introduced by Koß et al. (2012); Duchon et al. (2014), our intention was from the beginning not to use only predefined data only, such as *.csv files with profiles of recorded data, which would be the conventional simulation approach. Instead, the aim is that SES components are directly integrated into a simulation environment. This allows that the simulation works directly on real measurements that are communicated from external systems, similar to a hardware-in-the-loop environment. Therefore, we analysed the available tools and libraries and created a co-simulation framework that implements dedicated communication interfaces on top of different power flow solvers and enables the possibility to interact with external systems. We describe it in more detail in Chapter 4.

[8] `https://mosaik.offis.de/`, last accessed in March 2020

Our co-simulation framework helps us to achieve two goals: Firstly, we can quickly develop and simulate custom SES solutions and test their feasibility. Secondly, we can test real devices or applications that support industrial communication protocols, such as OPC UA, IEC 60870-5-104, Modbus or Siemens S7, and also web based APIs based on REST that are provided by other research departments with a context given by the simulated environment. In contrast to co-simulations like MOSAIK, the focus is therefore, not on the generation of large scenarios, but rather on the evaluation of specific control strategies and the definition of interfaces and flexible SES architectures, which we present in Chapter 5 and Chapter 6.

2.2.3 Further Technical Frameworks for Distributed Systems

Finally, we like to mention another approach that is not SES specific, but has relevance for this thesis and also today's systems in general, since it supports the implementation of highly distributed systems with available technical frameworks. The approach is similar to multi-agent systems, but it is based on the so-called actor model. The actor model was first described by Hewitt et al. (1973) and later extended with Hewitt and Baker (1977) and Agha (1985). In general an actor is a concurrent entity of computation. It has a message box, internal states and a behaviour that is triggered by messages. Actors interact by sending messages asynchronously. After receiving a message in its message box, an actor processes one message at a time. It executes the following actions concurrently in one step (Agha, 1985): (i) send a finite number of messages to other actors, (ii) create a finite number of new actors and (iii) determine a behaviour that handles the next message. An illustrative representation of the basic actor semantics is given by (Karmani et al., 2009) in Figure 2.10.

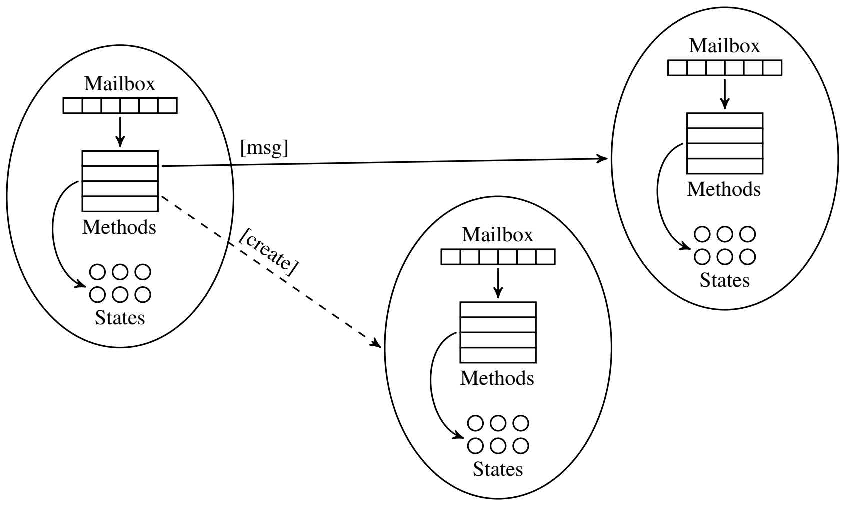


Figure 2.10: Illustration of the actor model (adapted from Karmani et al. (2009)).

The actor model has an impact for implementing systems such as web-services, objects with locks and functional programming, not only because it allows to implement highly distributed systems on different machines and distribute the program over the network, but also because it greatly facilitates to use multi-threading and modern multi-core CPUs with less dead locks and data races. This provides good scalability, where a single machine might need to handle thousands of requests per second. The scalability is not only useful to create web-services, but also implement simulations that include many different interactive components. The actor model has today strong support with available software libraries that greatly support implementation activities, such as scala actors (Haller and Odersky, 2009) and its successor library `akka`[9] as described by Haller (2012) and the documentation written and maintained by Lightbend (2018).

Relation to our work. The design of the generic architecture is not dependent on the introduced actor model. But the presented actor frameworks, particularly the recent implementations of akka (Haller, 2012) and (Lightbend, 2018) are helpful to develop a highly scalable framework to enable simulations and integrations to other external systems. The reason is that distributed, remote components should interact only via messages, and the desired simulation framework should be scalable. Our development efforts to establish a proper SES simulation framework uses the mentioned framework. More details about our approach and the used actor technology is described in Chapter 4, while concrete SES applications that are based on that technology are described in Chapter 5 and Chapter 6.

[9]`https://akka.io/`, last accessed in August, 2019

3 | Background and Foundation

In this chapter, we introduce basic terms, mathematical models and concepts that we use throughout the thesis. The chapter is structured into two main parts. Section 3.1 describes the necessary background for this thesis from the energy modelling perspective. It introduces relevant mathematics models that are used in current operation and their field of application. In Section 3.2, we discuss how those models affect the design of hierarchic architectures and related system interfaces. We focus particularly on EMS and coordination approaches in hierarchical structures. These two parts are the central aspects of this thesis and serve as our foundation to develop and evaluate the hierarchic SES architecture later on.

3.1 Energy System Models

The operation and control of SES rely heavily on mathematical models of power system engineers. Simultaneously, software engineers that create control and monitoring systems are faced with questions for designing appropriate system interfaces, since in practise, control and optimization systems require data to be transferred to be able to operate. The data specifications required by the interfaces are implicitly available within the mathematical models. Therefore, understanding those models as the hearth of system operation is of paramount importance to create SES applications and related architectures.

Mathematical models are included in different energy systems: inside control applications, inside scheduling and optimization systems, as a foundation for energy markets that enforce an economical operation, integrated in design tools to layout the infrastructure in order to configure and size the installed equipment, and of course to calculate costs, income or determine expected emissions. Depending on the purpose, such models vary in their granularity and associated details. The details depend on great extend on the systems at hand and on the time period that is of interest ranging from seconds, where models are the basis for functionality like control or safety, over minutes and hours, where models focus on operation and coordination of systems, up to months or even years, where models investigate system sizes, costs, numbers of components as well parameters and the types of required systems, as shown in Figure 3.1.

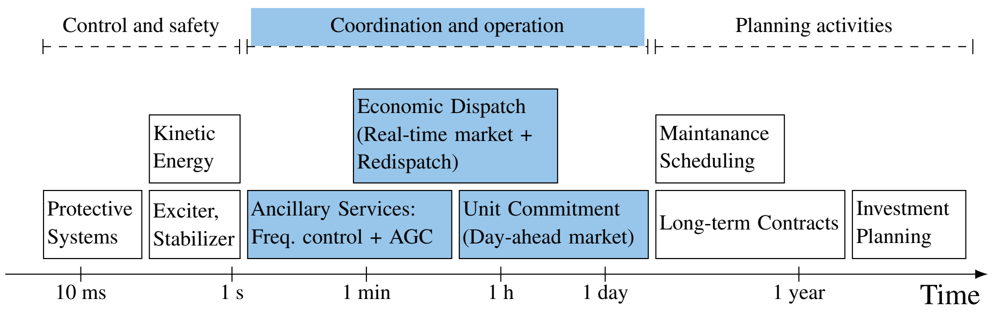


Figure 3.1: Key functionalities and related time scales for power system coordination and operation (adopted illustration from Jokić (2007) and Huber (2017)).

The focus of this thesis is the development of system architectures that enable the interconnection of EMS. We assume that EMS use conventional communication, such as TCP/IP. The reason is that many EMS are privately owned and operated. For communication, they rely on gateways that establish secure communications over the internet. Dedicated real time communication networks are probably to expensive for consumer EMS. Therefore, we focus on the coordination and operation time scales that are marked in blue in Figure 3.1. Fast control applications with real time requirements are out of scope.

Coordination applications cover three major tasks: unit commitment, economic dispatch and ancillary services (see also Huber (2017)). These tasks are supported by coordination approaches based on models that are economically driven, such as day-ahead markets and real-time markets, or based on models that consider technical limitations, such as re-dispatch measures by the system operator. Their main goal is to achieve a reliable, but economic power supply. Unit commitment tasks schedule the on/off-line status of large power plants, like coal or nuclear. Since those units have considerable start-up or shut-down times, this task is done quite beforehand. Economic dispatch determines the exact power production of each generation unit for the upcoming time periods. This is usually done quite close to operation. Ancillary services is a term that comprises all services that are activated at runtime to ensure a balanced system. It is a cascading system of control systems. It includes frequency control (primary control) that limit the deviations from the nominal frequency and automatic generation control (AGC) that restores the used primary control resources (secondary and tertiary control). Also those services are mathematically modelled and traded on particular energy markets.

The white marked blocks of Figure 3.1 indicate that there are many more systems required for power system operation, both on the faster and slower time scales which are not in the scope of this thesis. The blocks labelled protective systems, exciter, kinetic energy are necessary to keep the power system infrastructure technically intact and stable providing thermal protection or isolation, voltage stabilization, fast frequency drops, respectively. We refer to classical power system literature, e.g. Kundur et al. (1994) or Andersson (2012), for the interested reader to study those systems in more detail. The blocks labelled maintenance, long-term contracts and investment planning, belong to the area of risk management and infrastructure expansion optimization. We refer to energy system planning and energy economics literature, e.g. Kirschen and Strabac (2004) or Söder (2011), to get more inside into those activities.

There is no single, commonly accepted model, or approach for coordination that is used worldwide for all power systems, as power supply systems strongly depend on national regulations and market designs. The coordination approaches vary more or less from country to country, depending on the responsibility of power system stakeholders. An attempt to present the large variety of different possible mathematical models does not serve the purpose of this thesis. Therefore, in the following of this section, we introduce only a selection of models to demonstrate how they can be integrated into the desired architecture later on. We use a top-down approach. We start with high-level, abstract models of the market, which are responsible to improve the economical operation. We continue with more granular, physical models to present additional effects that effect the operation. Those models particularly address the operation of energy networks. Our investigation of energy system models is required to define appropriate system boundaries, related architectures and system interfaces later on.

3.1.1 Market Models

Today, the operation of energy systems is essentially coordinated by energy markets. Energy markets were introduced in the late second half of the 20th by the individual countries (Baldick et al., 2005), starting worldwide with USA and Chile, in Europe with England and Wales, and in late 1990s in Germany. The goal was to liberalize the monopolies of the state-owned or state-regulated, vertical integrated utility companies by splitting the responsibility between infrastructure operation (transmission and distribution), power generation and customers to facilitate market based competition. The intention was to create an open, transparent access for producers and (representatives of) consumers that participate at the energy system. The infrastructure operator was mostly decoupled from the market. He was only left with the responsibility of ensuring the safe and reliable operation of the infrastructure.

The energy supply system requires a balanced production (supply) and consumption (demand) of energy at any point of time. Otherwise, the system becomes unstable and blackouts occur. Energy markets support the balance with an economical mechanism to adjust supply and demand. They also ensure that producers are paid for the energy they generate, and consumers pay for the energy that they consume. Even if the power system is partly automated, the energy market and related payments are not performed at real-time[1]. As a solution, energy markets work with trading periods, for which producers and consumers have to provide bids. In principle, the periods can cover arbitrary length, but it has proven to be practical to use 1 hour and 15 min intervals (Kirschen and Strabac, 2004). Market players participate in the market by offering *bids* to purchase or sell energy. The bidding process is usually similar to an auction and differs depending on the desired national market design (Contreras et al., 2001). In Europe, we have the EPEX Spotmarket[2]. Here, market participants have quite a “simple” bidding process. They offer defined bids, which are collected and used to determine a price for the upcoming time periods. A single bid requires at least the following information: **buy or sell** indication, desired **quantity in [MWh/period]** (volume), desired **price in [€/$_{MWh}$]**, **valid time period(s)** and of course the sender’s **identifier** for instance the company’s name. Bids that cover multiple periods indicate that the offer must be fulfilled during those time periods.

Market price building is carried out after all bids have been collected. It starts with sorting. First, all purchase bids (demand bids) are sorted in a list with descending prices. Selling bids (supply bids) are sorted in a list with ascending prices. The sorted lists are denoted as a Merit-Order, since in a perfect market (many participants, everybody has the same knowledge) it reflects the marginal costs of the energy generation units, where units with the lowest cost are the first ones that are taken for production, while units with the highest cost are the last ones. After the ordering, the list with supply bids is contrasted with the ordered list of demand bids. This corresponds to a **quantity-price graph** for each future time period, as shown in Figure 3.2. The intersection yields the market-clearing price for that time period. Mathematically this point has a special meaning. As shown by Kirschen and Strabac (2004) and Söder (2011) the global welfare (the economic surplus of consumers and producers) is maximized at this point. This point is also

[1]There are however ideas to establish such mechanisms. Blockchains are investigated to technologically ensure related contracts and automate the verified payment processes.

[2]`https://www.epexspot.com`, last accessed in March 2020.

economically Pareto efficient, which means that deviations from marginal costs bidding reduces the income of that producer that deviates, while other producers benefit (under the assumption of a perfect competition, where single players have no dominating market power). All demand bids and all supply bids left of the point are accepted, borderline bids up to the point are partly accepted, the remaining bids right of the point are not accepted. In consequence, this market price leads to a schedule for power plants.

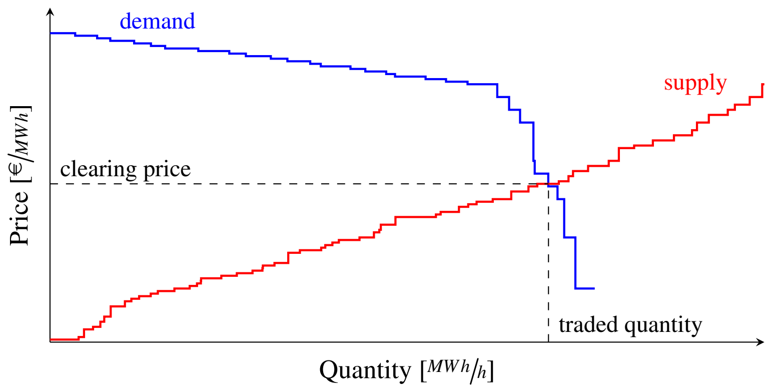


Figure 3.2: Supply and demand bids at some time period in a market clearing process.

The introduced market model represents the EPEX Spotmarket. It is well suited for day-ahead markets and intraday markets. However, as mentioned earlier other market models exist. For instance, ancillary services (operating reserves) are traded on a pay-as-bid markets (Müsgens et al., 2014). In Germany, the regulation authority Bundesnetzagentur introduced in 2018 a pay-as-bid procedure that consists of two parts, one for the compensation of standby duty based on power capacity and one for the actually delivered power (Bundesnetzagentur, 2018). This example shows that different market models might even coexist. Therefore, the introduced model is just a representation for coordination models that use prices. The approach relies on a demand, supply and a price specification and accurate forecasts, particularly for the demand and supply from renewable, volatile energies.

Using this model for coordination in a system with a hierarchic architecture that consists of different levels of EMS is possible, if the interfaces of the participating system are specified accordingly. Therefore, we can transfer this coordination approach to hierarchical arrangements of EMS components, but we have to consider several challenges. One such challenge is the specification of the right data for the interface. Another challenge is the consideration of forecasts[3]. We provide a deeper discussion about the necessary interface specification and hierarchic composition of the models further below.

[3]Note, forecasts become more inaccurate for smaller systems due to larger influences of single users that have a more stochastic behaviour than areas or districts (Esslinger and Witzmann, 2012). Therefore, in context to this work, investigations were done by Rottondi et al. (2015) and Bajpai (2018) to provide more accurate forecasts for EMS using time series analysis and machine learning approaches, respectively. Another possibility to handle forecast inaccuracies is to reduce forecast time horizons and approach real time energy trading scenarios. In context to this work, Thut (2018) used our technology (cf. Section 4.3) and investigated the technical feasibility of blockchains (Ethereum) for that scenario and explored related limitations.

Transmission Capacities

Market models rely on the assumption that energy can be transferred to any location without any restrictions. But this assumption requires a lossless and unrestricted flow of electrical power all over the system. In real systems, power is transferred over physical lines. They have losses and a maximum capacity, which might lead to transmission bottlenecks. Hence, for operation, market models need to be complemented with approaches, which consider those additional constraints.

Mathematically, transmission can be described by power flows that depend on different regions, their internal demand and supply that produces a power injection into the network, and available network connections between those regions. In real systems, the power flow is only determined by the fundamental laws of physics, i.e. Kirchhoff's laws. We describe these models in Section 3.1.2 in more detail. Here, we introduce a simplified version for coordination that considers transmission but in a more simple way, where only the first Kirchhoff's law is considered and the second law is neglected. This simplification is sometimes used in power system modelling to study costs for grid extensions, e.g. Schaber et al. (2012). It also gives us more insight in possible coordination scenarios, where the power flow can be controlled by operators.

To consider bottlenecks, energy system models introduce geographical regions that are interconnected. They represent system nodes. A geographical region is denoted with an identifier $i \in \mathbb{N}$. It has a demand D_i and generation G_i. When power is exchanged between different regions, we need to consider transmission lines with capacities. The regions inject power into the network with is transported over the lines. As a result, we can formulate a linear transport problem as follows. Assume that the region i has the demand D_i, the generation G_i and several energy exchanges $P_{k,i}$ over transmission lines with other areas. The first Kirchoff's law requires that the power is conserved in one node:

$$\forall i \in \mathbb{N}: \qquad 0 = G_i - D_i + \sum_{k \in \mathbb{N}} P_{k,i}, \tag{3.1}$$

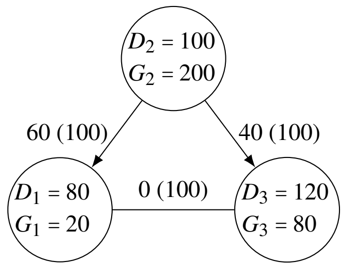
where $P_{k,i}$ denotes the incoming power flow from area k to area i. We can reformulate this representation into a set of linear equations and add constraints for the transmission capacities:

$$\mathbf{Ax} = \mathbf{b} \tag{3.2a}$$

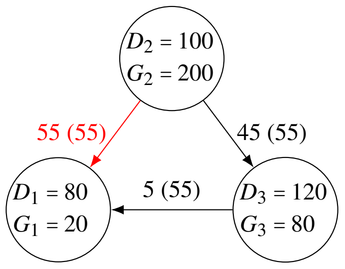
$$\mathbf{x_{lb}} \leq \mathbf{x} \leq \mathbf{x_{ub}}, \tag{3.2b}$$

where the vector $\mathbf{b}$ represents each region's internal power injection $G_i - D_i$, the vector $\mathbf{x}$ represents the power flows $P_{k,i}$, the matrix $\mathbf{A}$ represents if connections are available or not, and the vectors $\mathbf{x_{lb}}$ and $\mathbf{x_{ub}}$ represent the constraints due to transmission limitations. Note that $\mathbf{A}$ has generally not a full rang. It is therefore not invertible and multiple valid solutions exist. This is an important property of many realistic systems. It reflects the freedom to choose different possible solutions and opens the possibility for optimizations.

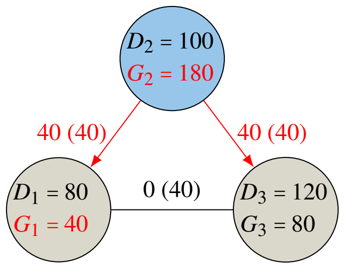
Example 1 To illustrate the work with transmission capacities we show an example of a system with three zones $1,2,3$ (see Figure 3.3). Each region has some demand D_i and generation G_i, which have been forecasted and determined by the market, respectively. We interconnect the regions with transmission lines that have capacity limitations, which are illustrated in ()-brackets. The numbers for the example are chosen arbitrarily for demonstration purpose.



(a) Possible power flows without relevant bottlenecks.



(b) Possible power flows with relevant bottlenecks and re-routing (if technically possible).



(c) Possible power flows with relevant bottlenecks, re-routing and re-dispatch. Two circle colours represent spiting into two market zones with different prices.

Figure 3.3: Coupling of different regions with transmission capacities in ()-brackets.

Depending on the limitations, different operation schemes in the network occur. As long as transmission capacities are large enough (Figure 3.3a), all generation and demand take place as determined by the market. When limitations constrain the operation (Figure 3.3b) the power transmission is re-routed. This is done by the system operator. For simplicity and just to demonstrate the effect of transmission limitations in this example, we assume that the power flow can be freely controlled by the system operator[4]. When all re-routing measures reach their technical limitations (Figure 3.3c) we wont have valid solutions for (3.2). In this case, power generation needs to be re-dispatched. Power plants in regions with missing power have to increase their generation, while power plants in regions with oversupply have to reduce their generation. The re-dispatching takes places as part of a congestion management process, for which different methods are available, as discussed by Söder (2011). As a result, some of the more expensive generation units are activated, which changes the marginal costs for the different regions and splits the price from the market perspective.

Equations (3.2) represent meshed networks, where also circular power flows are a possible mathematical solution. This is an undesired effect. To avoid solutions with circular power flows, we can reformulate (3.2) into a linear programming optimization problem, where we add costs for the transmission and write it as:

$$\min_{\mathbf{x}} \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{x} \tag{3.3a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.3b}$$

$$\mathbf{x_{lb}} \leq \mathbf{x} \leq \mathbf{x_{ub}}, \tag{3.3c}$$

here, $\boldsymbol{\lambda}$ represents the cost-vector for transmission. It is sufficient to add small positive costs, to obtain a solution without circular flows.

[4]Technically, this can only be achieved if the system operator has the right physical equipment to control the relevant physical properties, such as line impedances or phase angles, e.g. using modern FACTS (Flexible AC Transmission System) components as explained by Murali et al. (2010). However, we would like to stress that we cannot always rely on a freely controllable transmission (see also Section 3.1.2).

Regions and transmission capacities are used for planning and congestion management activities. The European Power System has multiple market regions and hierarchies, as shown in Figure 3.4. On the highest level, Europe has five regions that share a common synchronous electrical grid. The five regions are subdivided into more than 30 market areas, that are mostly aligned to countries, even though, some countries have several market areas (Italy and Nordic countries), while other countries (DE/AT/LU) share one market area. To further consider the transmission capacities, each market area can be internally subdivided into TSO (transmission system operator) areas, and these in turn can be subdivided into DSO (distribution system operators) areas. This forms a hierarchical structure of interconnected regions. Detailed analysis that investigate coordination approaches rely on different regions. For instance, the work carried out by Schaber et al. (2012) uses 83 regions for Europe, while Huber (2017) uses 268 regions.

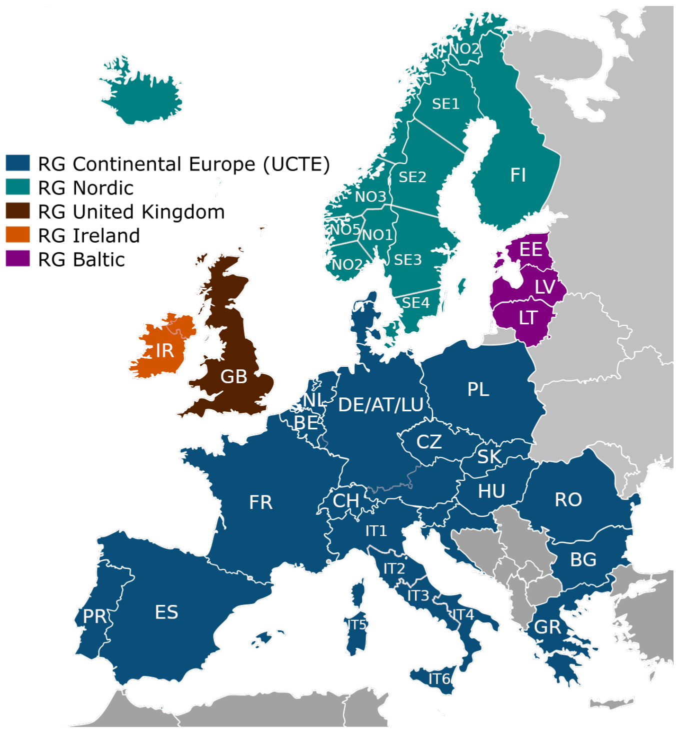


Figure 3.4: Map of the European Power System with its synchronous grids and market areas. (Illustration taken from Wikimedia Commons (2006) and adapted according ENTSO-E (2018))

Short-term Planning

The discussion of possible changes by coordination activities, such as re-dispach, leads quickly to further considerations that concern not only the network, but also the temporal level. For instance, unit commitment problems or economic dispatch (see e.g. Baldick (1995) or Huber (2017)), require additional considerations when turning on and off devices like power plants, due to start-up and shut-down considerations (time constraints and additional costs) as well as ramping rates. The temporal dependence is also important for storage systems. The possibility of charging and discharging devices affects the current and future power flows. Also storage capacities introduce further constraints. The consideration of such temporal dependencies is carried out by means of short-term planning activities that extend the models with transmission.

Models of power systems for short-term planning are often formulated as optimization problems (e.g. see Simoglou et al. (2010); Söder and Amelin (2011)). The objective is to determine detailed schedules that state how much each component should generate in the closest future. Normally, the short-term plan considers some defined time horizon, for instance one or two days. The plans are subdivided into a number of time periods. The choice of the time period length is arbitrary, but naturally a period length that corresponds to the time intervals used at the market is favoured. In Europe they correspond to one hour or 15 min intervals.

Short-time planning is applied for energy systems of various size (countries, cities, districts, large buildings) as shown by (Mancarella, 2014). Nowadays, more and more renewable energy systems, storage systems and low carbon technologies, like Combined Heat and Power (CHP), heat pumps, Power-to-X systems and electric vehicles are installed locally to improve the environmental footprint and save costs. This demands short-term planning problems to consider multiple energy networks at once. Therefore, power system optimization is extended with multi energy carrier systems, for instance as introduced in the two papers by Geidl et al. (2007); Geidl and Andersson (2007), and applied to different use cases by Orehounig et al. (2015), Liu and Mancarella (2016) and others. The idea of the model is to define energy conversion systems, called energy hubs, that are connected to different energy networks. An energy hub is multi-vector input/output relation that describes the interconnections of different energy forms (e.g. electricity, gas or district heating). The relation is characterized only through the consideration of power and the efficiencies of the converter devices. We use this approach to present a compact short-term planning model that follows the energy hub approach[5].

We use a linear optimization model, to illustrate how short-term planning models are defined (see also Chapter 6 for specific details). The general form is given by:

$$\min_{\mathbf{x}} \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{x} \tag{3.4a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.4b}$$

$$\mathbf{G}\mathbf{x} \leq \mathbf{h} \tag{3.4c}$$

$$\mathbf{x_{lb}} \leq \mathbf{x} \leq \mathbf{x_{ub}}, \tag{3.4d}$$

where the **x**-vector represents the input power that is subject to optimization, the **b**-vector the desired output power, the **A**-matrix contains the coupling factors (e.g. efficiency rates), the **G**-matrix and the **h**-vector describe the set of dependent temporal constraints, such as storage, while the vectors $\mathbf{x_{lb}}$ and $\mathbf{x_{ub}}$ vectors describe lower and upper limits for the input variable.

The major scheduling constraint in modelling energy systems is that the demand (the desired output) is covered by the production and storage operation (the scheduled input) at each time:

$$D(t) = \sum_{i=0}^{n} \eta_i P_i(t) + \sum_{j=0}^{m} S_j^D(t) - \sum_{j=0}^{m} S_j^C(t), \tag{3.5}$$

where for each time interval t, D represents the demand, P_i the production of unit i, η_i its efficiency rate, S_j^C and S_j^D the storage charging and discharging rates of storage component j, respectively. Further, production and storage rates are limited for each device, i.e.:

[5]We presented this model as a joint work with colleagues in Bytschkow et al. (2019).

$$0 \leq P_i(t) \leq P_{i,max} \tag{3.6}$$

$$0 \leq S_j^C(t) \leq S_{j,max}^C \tag{3.7}$$

$$0 \leq S_j^D(t) \leq S_{j,max}^D \tag{3.8}$$

where $P_{i,max}$ is the maximal power of device i, $S_{j,max}^C$ and $S_{j,max}^D$ the maximum charging and discharging rates of storage j. The capacity constraints of storage equipment is given by

$$-C_j^{t_0} \leq \Delta t \sum_t \eta_j^C S_j^C(t) - \Delta t \sum_t \frac{1}{\eta_j^D} S_j^D(t) \leq C_{j,max}, \tag{3.9}$$

where, $C_{j,max}$ is the storage capacity of storage component j, $-C_j^{t_0}$ the state of charge at the planning time t_0, η_j^C and η_j^D the charging and discharging efficiencies and Δt the time interval.

Further, in systems with multiple energy networks, we use labels to denote each network. For instance in a sector coupled district, we distinguish between *heat* power and *electrical* power. Furthermore, we consider discrete time (e.g. quarter-hour intervals) and a finite time horizon of the length N_H for the planning. Hence, the demand vector $\mathbf{b}$ is chosen as

$$\mathbf{b}^T = (D_{heat}(t_1), ..., D_{heat}(t_{N_H}), D_{elec}(t_1), ..., D_{elec}(t_{N_H})), \tag{3.10}$$

Similarly, we model the schedule vector with n production and m storage devices as[6]

$$\mathbf{x}^T = \left(P_1(t_1), \dots, P_1(t_{N_H}), \dots, P_n(t_1), \dots, P_n(t_{N_H}), S_1^C(t_1), \dots, S_1^C(t_{N_H}), \dots, S_m^D(t_1), \dots, S_m^D(t_{N_H})\right). \tag{3.11}$$

Matrix $\mathbf{A}$ represents conversion efficiencies. It is composed of several sub-matrices based on available components as

$$\mathbf{A} = \left(\mathbf{A}_1, \dots, \mathbf{A}_{n+m}\right). \tag{3.12}$$

Further, each sub-matrix has again sub-matrices that reflect the set of networks, as required by the demand-vector:

$$\mathbf{A}_k = \begin{pmatrix} \mathbf{A}_{k,heat} \\ \mathbf{A}_{k,elec} \end{pmatrix}. \tag{3.13}$$

Finally, the cost function is defined as a discretised integral over the time as:

$$Cost = \Delta t \sum_{k=1}^{N_H} \left(\sum_{i=1}^{n} c_i(t_k) P_i(t_k) + \sum_{j=1}^{m} c_j^D(t_k) S_j^D(t_k) + \sum_{j=1}^{m} c_j^C(t_k) S_j^C(t_k) \right). \tag{3.14}$$

It contains the costs for production and has the possibility to add costs for storage. This helps to avoid parallel charging and discharging, which is necessary for instance when thermal storage has costs for operation due to pumps. The cost vector is then:

$$\lambda^T = (c_1(t_1), ..., c_n(t_{N_H}), c_1^D(t_1), ..., c_m^D(t_{N_H}), c_1^C(t_1), ..., c_m^C(t_{N_H})). \tag{3.15}$$

[6]Note, in our model we consider charging and discharging a separate process, to have a normalized form of the optimization problem.

This set of equations defines the short-term problem, i.e. (3.4). The solution gives an optimal schedule for all components for the given time horizon. By employing the model for each time step iteratively while moving the time horizon, we obtain a model predictive control (MPC) approach, as introduced by Rawlings and Mayne (2009). It can be used to coordinate multiple components inside an EMS, or a microgrid. When the right information is available at an EMS interface, the MPC can be used to coordinate multiple EMS. We explain this possibility below.

The model can be extended in multiple ways quite easily. Firstly, we can add more networks. For instance, if multiple heating networks with different temperatures exist, we separately model those networks and the related converter components that transfer power from one to the other network.

Secondly, sometimes, for instance in EMS that are connected to a grid infrastructure, the demand is not only covered by internal production components. Some power is provided by the grid. Electrical power is a good example for that. The provision of power has costs that correspond to market prices. To model power delivery from the network and the power injection into the network we use the same formulation as for storage, where one sub-vector takes power and another one injects power. The market prices for taking and injecting are included in the corresponding λ entries. Note, our consideration of a time horizon enables to model variable prices.

Thirdly, the presented model assumes in Eq. (3.5) that there is only one demand $D_\xi(t)$ at some time per network ξ. Different regions and transmissions are not considered separately. To model transmission, we can extend Eqs. (3.5) - (3.13) with additional modifications. First, we split the demand according the individual regions, i.e. system nodes. We label those nodes with capital letters $A, B, \dots$. We obtain a modified output vector $\tilde{\mathbf{b}}^T = (\mathbf{b}_A^T, \mathbf{b}_B^T, \dots)$ that contains the individual demands for each region for each time for each network. Similarly, we adapt the input vector by allocating the production to the corresponding nodes. Now each element, e.g. $P_i(t)$, has now one additional label, e.g. $P_{i,A}(t)$ that includes its node relation. The conversion matrix is adapted correspondingly as well. Further, to consider transmission, the input vector is extended with power flows that might occur in the system, similar as in the transport problem defined by Eq. (3.3). To consider transport losses, we add a conversion factor as well. It means that if power is sent from, lets say, system A to system B, system A injects the power $P_{A,B}$ and system B receives only a reduced power described by $\eta_{A,B} P_{A,B}$, where $\eta_{A,B} \in [0..1]$. The inverted direction is modelled accordingly. We illustrate this in Figure 3.5.

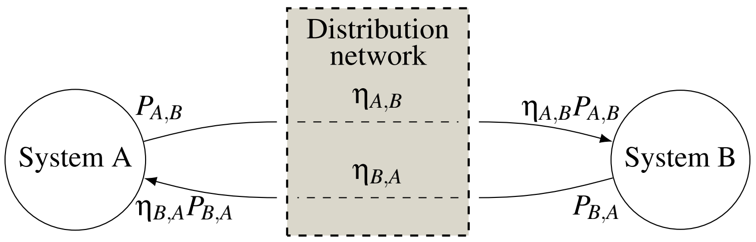


Figure 3.5: Transmission with a conversion factor.

Finally, we receive new vectors and matrices in the form of

$$\tilde{\mathbf{b}} = \underbrace{\begin{bmatrix} \mathbf{A}' & \mathbf{A}_{transmission} \end{bmatrix}}_{\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{x}' \\ \mathbf{x}_{transmission} \end{bmatrix}}_{\tilde{\mathbf{x}}}, \tag{3.16}$$

where $\mathbf{A}'$ and $\mathbf{x}'$ reflect the modifications due to additional node labels, $\tilde{\mathbf{x}}$ is extended by $\mathbf{x}_{transmission}$ that represents additional power flows and have to be determined by the optimization, and the coupling matrix $\tilde{\mathbf{A}}$ is extended by transmission lines and related efficiencies described by $\mathbf{A}_{transmission}$. For example, a system with a topology as illustrated in Figure 3.5 has the extensions

$$\mathbf{x}_{transmission} = \begin{bmatrix} P_{A,B} \\ P_{B,A} \end{bmatrix}, \qquad \text{and} \qquad \mathbf{A}_{transmission} = \begin{bmatrix} -1 & \eta_{B,A} \\ \eta_{A,B} & -1 \end{bmatrix},$$

where both values for power transmission $P_{A,B}$ and $P_{B,A}$ are ≥ 0. Note, the power transmission contributes to the demand in each region. Hence, we consider the first Kirchhoff's law (cf. equation 3.1). The optimization takes care that circular flows are avoided.

The presentation of this model is only one possible example to describe energy systems in the short-term planning context, which is important for the coordination of systems. Usually, those coordination activities are carried out by system operators with the goal of meeting demands at lowest costs, by utilities that maximize profits with their bidding strategies, or nowadays by EMS that control virtual power plants or microgrids. Often additional constraints have to be considered in the model, such as the possibility to have on-line and off-line devices that have a maximum and a minimum operational limit. In this case, the model is extended to a mixed-integer linear programming (MILP) approach, (e.g. Simoglou et al. (2010); Mancarella (2014); Huber (2017)). Some models consider additionally non-linear dependencies, for instance in the price, in their equality or inequality constraints, e.g. Geidl and Andersson (2007). This opens the path for the optimization algorithms in various directions. We do not extend the models much further, since this thesis focuses more on the integration of such models into a coordinated EMS control structure.

3.1.2 Physical Models

So far we have described models that focus on the coordination of energy systems based on steady state power flows using network models, where the energy transport is determined by the first Kirchhoff's law. This formulation allows to choose the direction of the flows quite arbitrarily, as long as the nodes fulfil the requirement of the first law. In real systems, however, the transport is not only determined by balanced nodes, but also by the second Kirchhoff's law in combination with resistivity for electrical networks (Ohm's law), and hydraulics including liquid flows, pressure, pressure-losses (Continuity equations, Bernoulli's principle, Darcy-Weisbach equations) for district heating. Since, these relations are specific for each network, no generic model that covers all types of power flow is available as shown by Geidl and Andersson (2007). In the following, we shortly summarize the two major networks for power transport that concern EMS, to explain the influence on the coordination approaches in the following.

Electrical System

Models of electrical systems are based on three basic laws: Ohm's law, which relates voltages and currents, and the two Kirchhoff's laws, which simplify the calculations in networks, by describing that in nodes, the sum of currents is zero, in closed loops, the sum of voltage differences is zero. Those three laws determine the power flow in the network.

The electrical power in a system is calculated in its general form with complex notation. By using Ohm's law and Kirchhoff in its complex notation, and then identifying the corresponding coefficients (for the derivation we refer to Andersson (2012)), the general power expression with node identifiers $k, m \in \Omega$, where Ω represents all system nodes, is derived as:

$$P_{km} = U_k^2 G_{km} - U_k U_m (G_{km} \cos(\theta_k - \theta_m) - B_{km} \sin(\theta_k - \theta_m)) \tag{3.17a}$$

$$Q_{km} = -U_k^2 (B_{km} + b_{km}^{sh}) + U_k U_m (B_{km} \cos(\theta_k - \theta_m) - G_{km} \sin(\theta_k - \theta_m)), \tag{3.17b}$$

where P_{km}, Q_{km} denote the active and reactive power flows from node k to m; U_k,U_m the voltages at those nodes; G_{km}, B_{km} the conductance and susceptance of the physical lines; θ_k, θ_m the phase angles at the corresponding nodes. A derivation of the network equations yield the corresponding injections at the nodes as:

$$P_k = U_k \sum_{m \in \Omega} U_m (G_{km} \cos(\theta_k - \theta_m) + B_{km} \sin(\theta_k - \theta_m)) \tag{3.18a}$$

$$Q_k = U_k \sum_{m \in \Omega} U_m (G_{km} \sin(\theta_k - \theta_m) - B_{km} \cos(\theta_k - \theta_m)). \tag{3.18b}$$

Using the matrix notation the injections are also written as:

$$\mathbf{S} = diag(\mathbf{U})\mathbf{I}^*, \tag{3.19}$$

where $\mathbf{S} = \mathbf{P} + i\mathbf{Q}$ is the complex power, $diag(\mathbf{U})$ a diagonal matrix made from the voltage vector and $\mathbf{I}^*$ the complex conjugate of the current vector. The current vector is again calculated from the voltage vector as:

$$\mathbf{I} = \mathbf{Y}\mathbf{U}, \tag{3.20}$$

where $\mathbf{Y}$ is the complex admittance matrix. Solving the equation (3.19) is a non-linear problem. Therefore, two very well known approaches exist that we summarize in the following.

DC Model: The first approach to solve the problem is using approximations that linearise the equations. The following simplifications are made: The voltage differences are assumed to be small and a per unit system is used, hence $U_k \approx U_m = 1$p.u.. The angle differences are small in a light load conditions, hence, $\sin(\theta_k - \theta_m) \approx \theta_k - \theta_m$. The susceptance is larger than the conductance, hence $G \approx 0$. The reactive power is neglected. Using the simplifications in equations (3.17), we obtain:

$$P_{km} = B_{km}(\theta_k - \theta_m), \qquad \forall k \in \Omega, \tag{3.21}$$

where the power flow depends only the phase at each node and the susceptance of the line. This leads to the DC model that is written for all nodes in matrix notation as:

$$\mathbf{P} = \mathbf{B}\theta, \tag{3.22}$$

where **B** is the nodal matrix of susceptances. The elements of that matrix are proportionally to the line length (Andersson, 2012; Huber, 2017). Note, **B** has not full rang and multiple solutions exist, in particular because the differences of the angles are of interest not some specific angles. To obtain a unique solution, one node is defined as a slack node, which has a fixed reference angle, e.g. equal to zero ($\theta_{ref} = 0$). This allows to leave out one line of the matrix and obtain a new, invertible matrix $\mathbf{B}'$, so that the remaining angles are calculated as $\theta = \mathbf{B}'^{-1}\mathbf{P}$, and with that the power flows in the system.

AC Model: In contrast to the DC model, the AC model does not use simplifications. Instead, the equations (3.19) are solved directly. The main work for the AC model is therefore to find an algorithm that allows to calculate a solution in a robust and quick way. There are a number of possibilities to do that. A popular approach is to use the numerical Newton-Raphson method, which was developed in the 1960's for the power flow problem by Carpentier (1962) and Tinney and Hart (1967). Therefore, equation (3.19) is rewritten as:

$$0 = diag(\mathbf{U})\mathbf{I}^* - \mathbf{S} = \mathbf{f}(\mathbf{U}), \tag{3.23}$$

with the non-linear function $\mathbf{f} \in \mathbb{C}^n$ that depends only on the variables $\mathbf{U} \in \mathbb{C}^n$. Then the solution is found iteratively by calculating a new voltage vector (including angles), that fits as closely as possible to the desired power injections of the nodes:

$$\mathbf{U}_{k+1} = \mathbf{U}_k - J_\mathbf{f}(\mathbf{U}_k)^{-1} \cdot \mathbf{f}(\mathbf{U}_k), \tag{3.24}$$

where the $\mathbf{U_k}$ represents the k-th iteration of the voltage vector, $J_\mathbf{f}(\mathbf{U}_k)^{-1}$ the inverse of the Jacobian matrix of **f** evaluated at $\mathbf{U_k}$ and $\mathbf{f}(\mathbf{U}_k)$ the function evaluated at $\mathbf{U_k}$. When $|\mathbf{f}| < \varepsilon$ the iteration might stop and the solution vector with the voltage is used to calculate the power flows, e.g. using equations (3.17).

It is a research domain to find accurate models and quick solvers to determine optimal power flows, as for instance described by Milano (2008). The major application area is to identify congestions and manage them by either a proper control of power flow (see e.g. Murali et al. (2010)), or to re-dispatch the power production (or consumption) when such options are available.

Summary: Both models, DC and AC, model the power transmission based on physical laws considering transmission line properties. The DC model is a simplification that neglects losses, voltages and reactive flows. But, it is fast in the solving process. Even if the losses are neglected, they can be reasonable approximated in a DC model, once the power flow is known (Andersson, 2012). Solving the DC model with the sum of load and losses gives a rather good solution for real transmissions, as shown in the study by Overbye et al. (2004). On the other side, the AC model considers a more granular level of detail, but requires more computational power. It is used, if we are particularly interested in physical effects and the transmission limitations related to them, for instance the identification of voltage instabilities. In general, however, the accuracy of both models, DC (load plus losses) and AC, is suited for congestion management as we obtain power flows that are observable in real systems (Overbye et al., 2004).

District Heating

The second network that is of interest for the coordination of EMS is district heating. It relies on a power transfer using hot water[7]. The ideal flow of water in pipes is described by the continuity equations (mass conservation) and Bernoulli's equations (energy conservation). Further, several losses are of interest, such as pressures losses (Darcy-Weisbach equation), temperature losses (depending on the pipe insulations, radius, temperature differences, pipe length, etc., e.g. as described by Nussbaumer et al. (2017)) and factors like tightness. For the coordination of EMS, we are interested in a model of the power transfer, similarly as for the electrical system. Therefore, we have to consider several mathematical relations.

First, the load at each house is determined by the heat exchanger, which connects the network (primary side (*pr.*)) and the house (secondary side (*sec.*)) as shown in Figure 3.6. The power transfer P at each substation is determined by energy conservation:

$$P = \dot{m}_{pr.} c_p (T^W_{pr.} - T^C_{pr.}) = \dot{m}_{sec.} c_p (T^W_{sec.} - T^C_{sec.}), \tag{3.25}$$

where $\dot{m}_{pr.}, \dot{m}_{sec.}$ are the mass flow rates at the primary and secondary side, c_p the specific heat capacity of the fluid, $T^W_{pr.}, T^W_{sec.}$ the warm temperatures and $T^C_{pr.}, T^C_{sec.}$ the cold temperatures.

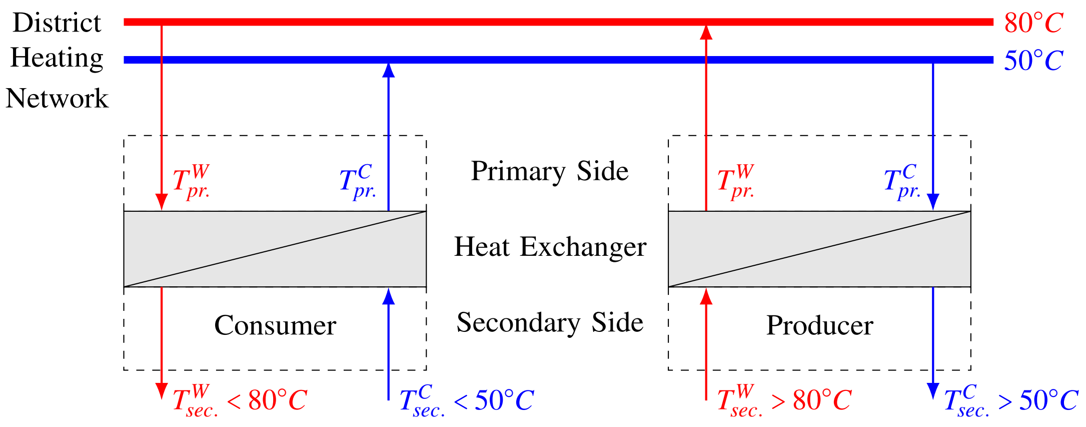


Figure 3.6: Two exemplary substations connected with district heating: The left system acts as a consumer, the right system acts as a producer. Given some network temperatures for the operation, the substations will have certain temperature limitations on their secondary sides.

To establish some desired power exchange, we have several possibilities. We can control the mass flows $\dot{m}_{sec.}$ or $\dot{m}_{pr.}$ with pumps or valves, depending on the available systems. Those values and the internal temperatures provide the corresponding conditions for the possible power exchange. Further, the mass flow $\dot{m}_{pr.}$ is a measure for the power transfer, but it is also part of the network flow balance and important for the hydraulics. With the volume flow $Q_m = \dot{m}_{pr.}/\rho$ the flow balance can be described for an arbitrary node m as:

$$Q_m - \sum_{n \in \Omega} Q_{mn} = 0, \tag{3.26}$$

[7]Initially, also steam based heating networks were available, but nowadays, most of the district heating relies on hot water (Lund et al., 2014).

where Q_m is the volume flow injected at node m and Q_{mn} the volume flow to the connected nodes in the network. The volume flow is physically created by the pressure differences and the pipe characteristics, which add resistivity to the flow. The relation of the volume flow and pressure is derived from the Darcy-Weisbach equation, e.g. described in Sigloch (2017):

$$\Delta p_{mn} = p_m - p_n \propto Q_{mn}^2, \tag{3.27}$$

which can be reformulated into a square root relation of Q_{mn} as:

$$Q_{mn} = k_{mn} s_{mn} \sqrt{s_{mn}(p_m - p_n)}. \tag{3.28}$$

where, k_{mn} is a constant factor that depends on the pipe characteristics (see also Sigloch (2017); Nussbaumer et al. (2017)) , p_m, p_n the pressures at the different nodes and

$$s_{mn} = \begin{cases} 1 & \text{if } p_m > p_n \\ -1 & \text{else.} \end{cases} \tag{3.29}$$

District heating has several additional facts that are important to know for coordination. To have a stable hydraulic system (i.e. pressure within the specified limits), the sum of all mass flows $\dot{m}_{pr.,m}$ adds to zero, i.e. $\sum_{m\in\Omega} \dot{m}_{pr.,m} = 0$. Further, the power transfer is proportional to the mass flow and the temperature difference between the cold and hot side of the network: $P_m \propto \dot{m}_{pr.,m} \cdot \Delta T_{pr.,m}$. Therefore, to avoid hydraulic problems and the mixing of the temperatures that destroys enthalpy and reduces the efficiency, **it is reasonable to demand to operate the network so that all power injections correspond to a proportional mass injection.** The temperature difference between $T^W_{pr.}$ and $T^C_{pr.}$ should be kept as constant as possible, similarly to the frequency in electrical networks.

Finally, the loss of temperature depends on many factors, such as pipe radius, pipe's heat transfer coefficient, heat capacity of water, external temperatures, volume flows and the line length leading to an exponential expression (see also Glück (1984); Nussbaumer et al. (2017)):

$$\Delta T = \left(T^0 - T^{env.}\right)\left[1 - e^{-c \cdot \frac{L}{\dot{m}}}\right], \tag{3.30}$$

where c is a factor that considers pipe parameters and medium characteristics, $T^0, T^{env.}$ the network and the environment temperature, L the line segment length, and $\dot{m}$ the mass flow between the nodes.

Note, the combined model with equations (3.28) leads to a non-linear, non-convex model (Geidl and Andersson, 2007). Hence, we cannot combine the electric and district heating networks in terms of their power injections that easily into a common optimization framework that works only with power injections. However, for the coordination we can work with approximations (see Section 3.1.1). We further observe that the energy losses that depend on temperature losses are smaller with higher mass flows. At the same time a high mass flows requires more pressure differences that increases the power required by the pump. The design of a suitable coordination will therefore require a trade-off between a high mass flow and the energy losses.

3.2 Coordination Systems

The presented models are designed to describe energy systems as exactly as possible. They allow to carry out different systematic studies and ultimately design coordination approaches in order to optimize the system operation. For that, the developed models assume that some system - *generally a software system* - collects all the required data (e.g. Huber (2017)). After the data collection, the software system calculates the best possible operational set-points and reports them back to the components. Such systems are for instance coordination systems of independent system operators (ISOs) or utilities. This assumption, however, faces several obstacles that cannot be solved that easily in reality.

One obstacle is that the data is not available. There are several reasons for this. One reason is that most of the models simply assume that they will receive the necessary information. They only define a system boundary from the model's perspective, but they omit to define a system boundary from the software system's perspective. Therefore, it often remains unclear, which of the systems (devices or EMS) have to be connected. Also specification of the data at that boundary, particularly the required level of detail is often left open. That's why we do not see any general system interface specifications that can be used for coordination. Instead, we only observe manufacturer-specific solutions in real systems, including hardware boxes, software systems and data interfaces (e.g. system integrators such as Next Kraftwerke GmbH). These boxes are send to the customer and directly connected to the local infrastructure. This opens of course a range of possibilities for monitoring and control, but it does not help to specify clear generic system boundaries, or to improve the interoperability. In addition to interoperability, it is also unclear if additional systems shall be connected and provide necessary data from external systems, such as forecasts in SES. Shall they be part of local devices or EMS? Shall they be part of the coordination system itself, or of some intermediate systems? Such questions remain unclear, as long as no explicit system boundaries, i.e. interface definitions are provided for software systems.

Another obstacle for coordination is that we have different systems for monitoring and control on different hierarchical levels. On the one hand, we have local control systems (e.g. PLCs (Programmable Logic Controllers)) that consider local boundaries and operational constraints of single devices to ensure a reliable operation. On the other hand, we have systems that integrate and coordinate multiple devices to simplify the operation or reach a more optimized operation. For instance, we might have an EMS that combines the heating system, with a heat storage, CHPs, photovoltaic and building automation to meet the buildings demand. On top of that, we might have an EMS to coordinate power supply in local quarters. On top of that, there might be a VPP that interacts with the market and monetarizes the residual energy of the quarters. It is obvious, that not every data point needs to be communicated from PLCs upwards to the VPP. It is also obvious that not all VPP data are of interest for local PLCs. Instead, we are interested in available options to aggregate data upwards and in options to break down the signals from the upper system into a coordination signal for the individual subsystems. Knowing these options allows to establish coordination more easier.

In the following of this section we address the two obstacles from above. We describe a suitable system boundary to enable the data collection and its transfer into the desired models in the first step. In the second step, we describe how these data are used in hierarchical systems.

3.2.1 System Boundaries

The introduced models for coordination require several input data, like the current demand and production, forecasts for demand and production (e.g. solar), specifications of production devices that can be controlled (e.g. CHPs, storage), prices or CO_2 emissions that arise when the devices operate, etc. The system that is described with those models is usually represented based on a geographically network topology with nodes and links between them, similarly as shown in Figure 3.7. All nodes belonging to the model are within the system. The system boundary of the model is in this case a line around an area that includes all contributing nodes.

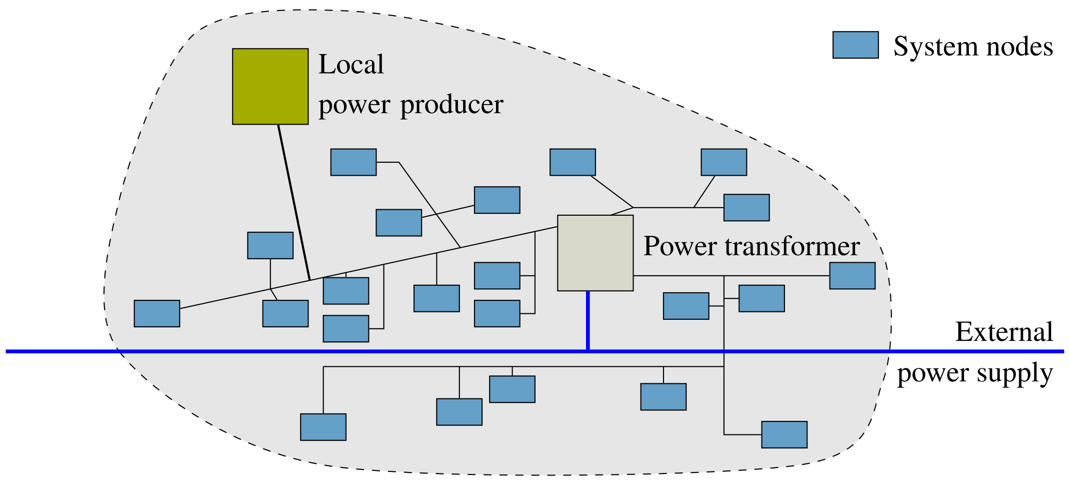


Figure 3.7: A system boundary for coordination from a model perspective. (Own illustration adapted from Wolff and Jagnow (2011))

In contrast to that, to reach the desired coordination we need a mechanism that collects the data and provides an coordination signal to all systems that participate in the coordination process. Such a mechanism can for instance be implemented with a coordination system (in general a software) of an ISO or a utility. This requires to define a system boundary of the software based coordination system, but also for those systems that participate in the coordination process. Software systems interact with their environment only by receiving and sending data. The system boundary is defined by their interface that specifies the received **Input** and generated **Output** (Broy and Stølen, 2001). An exemplary interface is shown in Figure 3.8.

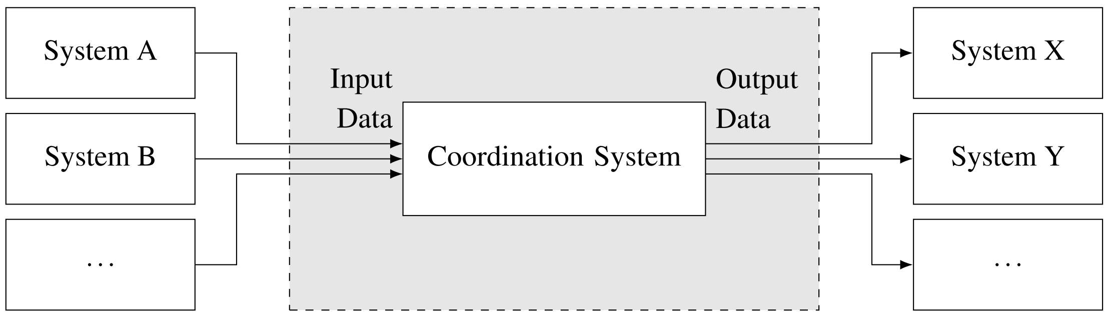


Figure 3.8: A system boundary for coordination from a software perspective.

To rely on system wide coordination activities with mathematical models, similarly as presented in Section 3.1, we need to define a vendor-independent system interface of the coordination system. It allows to express what we expect from the connected systems. To illustrate one example we take the first of the presented models (market based coordination from Section 3.1.1) and specify an interface of the coordination system in the following. Market coordination activities work with bids. Hence, they are the **input** for the coordination system. The **output** is the system price and the notifications if the bids have been accepted or not.

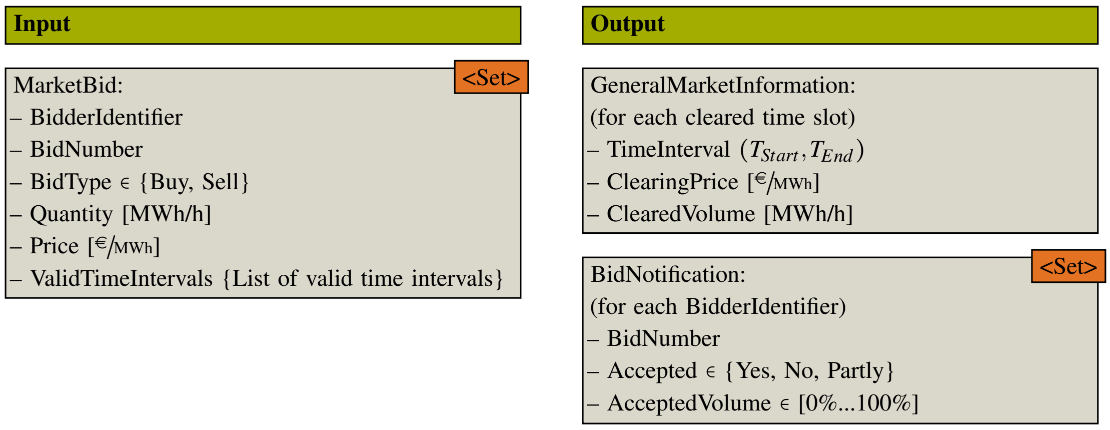

Input
<Set>
MarketBid:
– BidderIdentifier
– BidNumber
– BidType ∈ {Buy, Sell}
– Quantity [MWh/h]
– Price [€/MWh]
– ValidTimeIntervals {List of valid time intervals}
Output
GeneralMarketInformation:
(for each cleared time slot)
– TimeInterval (T_{Start}, T_{End})
– ClearingPrice [€/MWh]
– ClearedVolume [MWh/h]
<Set>
BidNotification:
(for each BidderIdentifier)
– BidNumber
– Accepted ∈ {Yes, No, Partly}
– AcceptedVolume ∈ [0%...100%]

Figure 3.9: Input / output specification for a coordination system based on market mechanisms.

The input / output specification can be illustrated as shown in Figure 3.9. The coordination system accepts `MarketBids` as input from the connected systems. They represent a legally binding offer for the coordination system. They contain the `BidderIdentifier` to assign the message to the sender, the `BidNumber` to differentiate between different bids from each sender, the `BidType` to decide if the sender desires to produce or consume energy, the `Price` and the `Quantity` as the main bid parameters, as well the `ValidTimeIntervals` to denote the time intervals for which the bid is valid.

The coordination system provides two types of data as output. Firstly, it provides the general market information for all connected systems. This includes a market price signal for each time slot and the cleared volume. Secondly, it provides individual information for each connected system. Since each bidding system is allowed to offer several bids, the individual information to each *BidderIdentifier* contains a set of acceptance notifications for the received `BidNumbers`. The output is a legally binding acceptance of the offered bids. The `BidNotification` states if the bid has been fully accepted (100%), fully declined (0%) or partly accepted (a value in the range 0%...100%) in accordance to the bid offering mechanism described in Section 3.1.1.

The specification of the interface describes what the coordination system expects from its environment and clarifies the required level of detail for the data exchange. From the specification point of view, we assume the coordination system cannot interact with its environment beside the defined system interface. This sharpens the boundary of the desired system and allows to carry out tests of the algorithms used by the coordination system. The specification, however, does not define yet how the data is exchanged. There are many creative ways to achieve such a data exchange. For instance, the data can be transmitted as a letter, as an email, an excel file attachment of an email, written into a database, send over a form via https or with the use of communication buses, MQTT protocols, OPC UA, etc. The specification does also not define, who sends what, i.e. the coordination system might expect to receive the data, but it can also actively request the data from the systems participating in the coordination process. Luckily, there are many different libraries available that help in implementing the different communication protocols. Therefore, the specification of the data is the most crucial part for the definition of the system interface and it fits well to the task of modelling energy systems. In this thesis, we specify interfaces for different models in more detail in Chapter 5 and Chapter 6.

3.2.2 Internal System Knowledge

Coordination systems rely mostly on received input data to perform their calculations. But for some tasks they might require additional system knowledge in form of the network topology and its physical properties (see Section 3.1.2). For instance, for congestion management, the coordination system requires to know the transmission lines between the system nodes, their capacities and properties that lead to losses. This knowledge has either to be "given" to the system, e.g. encoded by an engineer that models and parametrizes all connections, or it can be learned from (historical) nodal measurements that measure power injections and voltage values (e.g. as described by Deka et al. (2017)). The availability of internal system knowledge determines, which models can be used for coordination.

The knowledge about the network topology improves the coordination. The coordination system becomes more accurate, as physical effects like power losses can be calculated. The coordination system is also aware of undesired network states that lead to congestions. Therefore, it can apply congestions management (if the right inputs and outputs to control systems exist) by rescheduling the demand, supply or storage, controlling components that affect the power flow like switches, transformers, inverters, for electricity, or valves and pumps in district heating. Sometimes the right level of internal knowledge cannot be obtained. The reasons are country and sector specific regulations, lack of valid network information, or temporary contractual arrangements. For example, for electrical networks there is the so-called unbundling in Germany. Owners of the infrastructure (e.g. utilities) that have the network topology data are independent of service providers that interact with nodes (customers). From the coordination point of view, services providers should offer contracts with flexibilities and coordination possibilities. But they have not a good internal knowledge. In addition, due to liberalization, every customer is legally allowed to choose his service provider and even change its contract from time to time. Therefore, the ideal system with clear boundaries (Figure 3.7) that has a good knowledge of internal network topology is even harder to establish. In reality we observe systems that have only limited internal knowledge of a system, as illustrated exemplary in Figure 3.10. All these factors complicate the development of coordination systems. That's why we see two types of coordination systems with respect to the internal system knowledge.

One type of coordination systems work only with data from interfaces and have no internal knowledge about the topology. The goal of this type of system is to improve the economical operation of local power producers and consumers, by offering dynamic prices that depend on the market. Those can be used to optimize the production and demand. This type of coordination

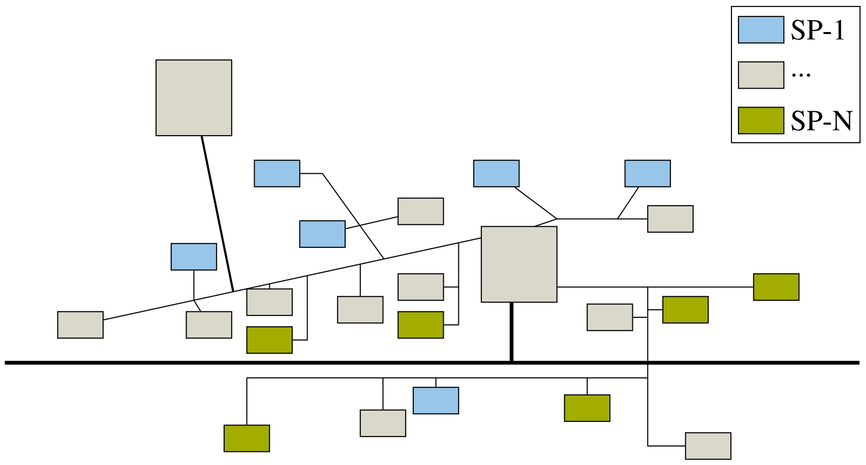


Figure 3.10: A network of nodes with different service providers (SPs).

systems is often denoted as VPP. The most important advantage of this system is its high flexibility to integrate other components. VPPs can be used for central coordination. In this case, a VPP is the only decision making system that sends control signals directly to the components. VPPs can also be used for decentralized coordination. In this case, a VPP determines only a guiding signal (e.g. a market price) and the components them-self decide how to schedule their supply or demand. We explain this type of coordination systems in Chapter 5 in more detail.

The other type of coordination systems has the authority in a clearly defined area. It works with data from interfaces combined with internal knowledge about the network topology to achieve a better coordination for a certain area. Such coordination systems match the power supply and demand while considering the power flows and losses. To do so, they integrate a critical mass of components that have an influence on the power flows. This type of coordination systems is often denoted as "microgrid". Preferably, the microgrid coordination system is the only decision maker (signal provider) in a certain area, to avoid contradictory signals that lead to unexpected results. We explain this type of coordination system in Chapter 6 in more detail.

3.2.3 Model Creation with Data from Interfaces

As we describe above, coordination systems are not omniscient, they rely on data that they obtain over their interfaces. They are also limited by their internal system knowledge. Depending on the chosen model for coordination, the coordination system needs to transfer the data from the interface representation into another form to solve the model and transfer it then back to send output data. Sometimes this is easy and straight forward. Sometimes the data processing requires significant expertise. In this section, we explain what a developer of coordination systems should be aware of when transferring data from one form into another one.

A coordination system receives input data from multiple sources, as illustrated by the system boundary in Figure 3.8. Mostly, the data originates from external systems that represent single devices, single buildings or sometimes even areas (e.g. microgrids with internal networks). The received data contains different kind of information. For coordination, the most important information includes the expected demand and the potential supply of all connected components.

This information has to be transferred into a suitable data form that depends on the chosen model. We present two brief examples in the following to show the differences of the data transfer. One example addresses the auctioning based model and another one the short time planning model as described in Section 3.1.

Auctioning Based Model

An auctioning based model works with bids (see Figure 3.2). The received data is similarly to `MarketBids` (as presented in Figure 3.9). This data is easy to process. The first step is to collect all bids and create two sets, one for demand (buying bids) and one for supply (selling bids). Then, both sets are sorted. The sorted sets are used to find the intersection point (the market clearing price). This point immediately separates the bids into those that are accepted and those that are declined. The price and the acceptance notification represent the coordination output. The development of such a coordination system is easy and straight forward. The reason is that the coordination system receives the data already in a the required form. The difficulty for data preprocessing is shifted to the connected EMS, which create their bids based on forecasts and available equipment. Some equipment types are easy to represent as bids, e.g. demand, others are not, e.g. storage. But this issue is a question for the connected EMS, not the for the developer of the coordination system.

Short-term Planning Model

The short term planning model uses vectors and matrices (equation 3.4). In difference to the auctioning based model the data is not directly available in the right form, because the connected EMS cannot send vectors and matrices. Instead, they send data representing their demand and supply in a parametrized way, saying which demand is expected and which devices with which parameters are available. Therefore significant data preprocessing is required to create the right vectors and matrices. It is reasonable to consider each model parameter individually to create a short term planning model. We briefly describe which data contributes to which model parameter in Table 3.1.

Table 3.1: Relation of received data and model parameters for the short time planning model.

Model parameter	Parameter details	Data required
$\mathbf{b}$	$(\mathbf{b}_{heat}, \mathbf{b}_{elec}, \ldots)^t$, where $\forall k:$ $\mathbf{b}_k = (b_1, \ldots, b_{N_H})^t$	The **b**-vector represents the expected demand. To create **b**, a demand forecast is required for the time horizon of length N_H and the considered networks. The creation of **b** depends on whether we have a model that considers transport or not. When transport is not part of the model, the demand is aggregated into one value. Otherwise, the demand-vector is node specific.

$\mathbf{x}$	$(\mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_{n+m})^t$	The $\mathbf{x}$-vector represents the desired operation of supply units, storage, externally connected networks and transport. The sub-vectors $\mathbf{x}_k$ represent generation units for $1 \leq k \leq n$. They have only one desired value for each time step and their length is N_H. The sub-vectors $\mathbf{x}_k$ for $n \leq k \leq n+n$ represent units that are able to provide and take power, such as storage or external networks. Their length is $2 \cdot N_H$. The reason for this split is to keep all elements of the vector ≥ 0. To create $\mathbf{x}$ we only require the information which units are available and N_H, because the values are determined with the optimization.
$\mathbf{A}$	$\left(\mathbf{A}_1, \ldots, \mathbf{A}_{n+m}\right)$, where $\forall k$: $\mathbf{A}_k = \begin{pmatrix} \mathbf{A}_{k,heat} \\ \mathbf{A}_{k,elec} \end{pmatrix}$	The conversion matrix $\mathbf{A}$ contains the efficiency factors for all units that are also modelled in $\mathbf{x}$. The sub-matrices $\mathbf{A}_k$ represent supply units for $1 \leq k \leq n$, as well as storage and external networks for $n \leq k \leq n+m$, due to the aforementioned reasons. Many supply units contribute to one particular network. In this case only the related sub-sub-matrix $\mathbf{A}_{k,network}$ has entries. The other sub-sub-matrix is zero. When units contribute to multiple networks all related sub-sub-matrices contain the efficiencies. This inextricably links power injections of multiple networks. We call those units “coupling units” to denote this linkage. Storage units and external networks provide or take power. The corresponding sub-matrices $\mathbf{A}_k$ have adapted dimensions. To create $\mathbf{A}$ we require the efficiency values of each unit and each network (see also equation 3.5).
$\mathbf{G}$	similar sub-structure as $\mathbf{A}$	The $\mathbf{G}$-matrix represents relations that are time dependent, such as the charging and discharging. It has the same sub-structure as matrix $\mathbf{A}$. If a unit has no dependencies related to time, the corresponding entrances are zero. To create $\mathbf{G}$ we require the efficiency factors of the storage as described in (3.9).
$\mathbf{h}$	similar sub-structure as $\mathbf{x}$	The $\mathbf{h}$-vector contains time related model constraints, for instance storage capacities. For $\mathbf{h}$ we require the current state of charge and the maximum capacities in order to calculate the remaining constraints (equation 3.9).
$\mathbf{x}_{lb}$, $\mathbf{x}_{ub}$	similar sub-structure as $\mathbf{x}$	The vectors $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$ represent the limitations of $\mathbf{x}$ (see equations 3.6-3.8). For non-volatile, fully controllable generation components those values are fixed and given by the component manufacturer. For volatile units, like wind or solar, weather forecasts determine $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$.

λ	similar sub-structure as **x**	The λ-vector represents the operational price for each unit. For most units, the price is fixed, but for the external suppliers (external network) the price might vary. To create λ we require the price information from each unit and the external supplier (e.g. market price).

Details of short time planning problems are very important for developers, who create the interfaces and develop the data processing. Even if the general model looks similar, the data processing might differ. To explain that, we can compare a simple model (no transmission, no network topology) and a similar looking model that covers one additional feature (transmission with network topology), where some of the vectors and matrices have to be adapted. Without a topology, the demand of multiple EMS is aggregated into one value. With a topology, the demand of each EMS is represented individually as a node in the model. With this modification, the general model (equations in 3.4) looks still similar, but the **b**-vector, the **A**-matrix and the **x**-vector are different (equation 3.16). The adapted model allocates the demand of each EMS to a particular network node that changes **b**, maps each unit to a node that changes also **A** and add transport links based on the network topology that changes **A** and **x** as well. Those details requires an adapted processing of the data. Depending on the extension, also the results (represented by the **x**-vector that are mapped to the output) might need a modified data processing. The remaining parts of the coordination system, like the interfaces, or the solvers are easy to be reused, since the model remains a linear programming problem in this case. Due to these details, it quickly becomes clear that a collaboration of power system engineers and software developers is essential for the specification of interfaces that describe which data is required for the models as well as the data processing related to those data specifications.

Additional Observations

Our development efforts (see Chapter 5 and Chapter 6) reveal additional helpful details that support the integration of models in coordination systems. The expectation that data is provided over interfaces and not as data files or scripts, which are often used in MATLAB, affects our model design. We expect that there might be different numbers of connected EMS. We also expect that all EMS contain individual equipment. Some EMS provide only demand data, some EMS provide only supply unit data, some EMS provide both and add storage data on top. Therefore, the model design needs to support the handling of data that contains different units, different parameters and different compositions.

This variety leads to a more modular design of the model. The elements in modular models are all structured in sub-vectors and sub-matrices that are created and modified step by step when parsing the input data. This allows us to collect initially all inputs of the connected EMS. After that, the data processing creates multiple sets from the data: one for the demand, one for controllable supply units, one for volatile supply units, one for couplers and one for the storage systems. Then we go through each set and modify the model accordingly.

Such a modular design is not only necessary to develop a coordination system, it is also very beneficial for planning tools, which focus on designing different scenarios with different sets of buildings and equipment. This allows to compare scenarios with different equipment, different model parameters (efficiencies, costs) analyse different time horizons and test different operation goals such as an economic or an emission based coordination. We therefore recommend modular model designs that are simply to extend not only for coordination systems, but for any kind of models related to EMS.

3.2.4 Hierarchical Structures

Continental power systems are interconnected. Nevertheless, system boundaries determine responsibilities and are therefore indispensable for operation. The interconnection of several power systems entails the need for coordination between them. This adds additional system boundaries that affect both geographic representations and software interfaces, as illustrated in Figure 3.11 and Figure 3.12. These boundaries and responsibilities form a hierarchical structure with several layers of components with nested system boundaries. To describe the hierarchy more convenient in the following, we use the terms children and parents to denote connected systems that belong the layer below and above, respectively.

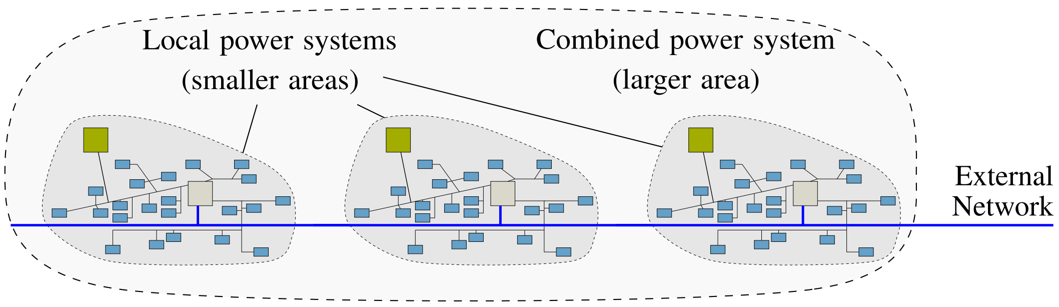


Figure 3.11: System boundaries for coordination in hierarchical systems: model perspective.

The geographical representation structures the different areas hierarchically. One large area is composed of several interconnected sub-areas. A sub-area is again composed of interconnected sub-systems, like other sub-areas or buildings. The composed entities determine some form of responsibility, like ownership or administration. This hierarchic composition fits very well to the general electric grid infrastructure. At the lowest level there are buildings connected to the low voltage network (e.g. 230 V/ 400 V) with a very broad spectrum of power supplies, ranging from very simple to very complex systems. Simple systems have only one fixed house connection with a single measuring device as their power supply. Complex systems have the same connection and in addition a wide range of controllable devices, such as photovoltaic, battery systems, CHPs, heat pumps, thermal storage, charge control for electric vehicles and controllable demand units. Such complex systems have their own EMS, which are the essential component to interact with coordination systems. On the next higher level we have districts or rural areas, which are connected to the medium voltage (e.g 1 kV/ 35 kV) over transformer stations and have power generation units such as wind parks and city power plants. These areas are similarly structured as shown in Figure 3.11. Currently, these areas have no dedicated EMS, but most of the renewable units are installed on that hierarchical level. Therefore, the question how to enable a better control in these areas with coordination systems is quite important for our future energy infrastructure design. Depending on how large the area is, it makes sense to divide it hierarchically into individual buildings, smaller-areas, medium-areas, larger-areas and so on in such way that each division has a clearly defined boundary for a coordination system. The next higher level is the transmission network with a high-voltage (≥ 110 kV). It supplies large regions or cities and covers long distances. Only the largest production units, like nuclear, coal or hydro power plants are directly connected to that network. All those units have their own EMS with dedicated interfaces, such as SCADA systems.

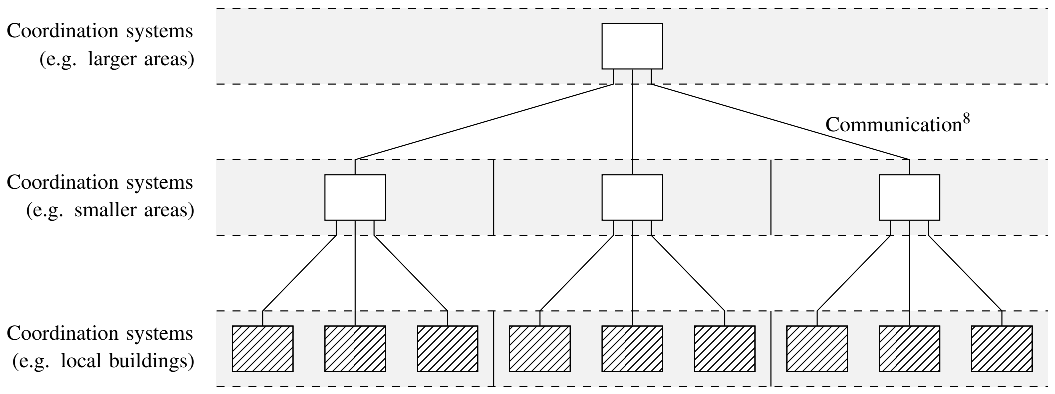


Figure 3.12: System boundaries for coordination in hierarchical systems: software perspective.

A similar hierarchic composition as we see for power system networks can be applied to coordination systems. The composition has to be software based. A composition of software systems is carried out via the interfaces as system boundaries. In a hierarchical system, a coordination system always interacts with other EMS, or in other words, with other coordination systems[9]. The connected EMS are either children representing systems of smaller areas or parents representing systems of larger areas. In a hierarchy the coordination system provides potentially an input / output interface for both adjacent layers. In addition, we do not know in general how many layers are desired. Therefore, the interface needs to be specified such that there can be arbitrary many intermediate layers. We can achieve that, when we demand that a coordination system is always a potential child system, which can be connected to a parent. The parent is only a coordinator. It interacts just with its direct children without knowing the "deepness" of the structure below, i.e. it interacts only via the interface without being aware of the system's composition. This simplifies greatly the system design and the coordination. But it requires a generic interface specification for the systems that participate in the coordination process. The specification needs to cover data that allows to calculate coordination signals based on a common mathematical approach, similar as we introduced in the previous section. Private information, such as descriptions and parametrization of internal devices, does not need to be exposed.

[8]The communication between two coordination systems is bi-directional. Each system offers a defined input and output interface for that. The interface specification describes which data can be received and which data can be send. In an implemented system, the interface specification includes further information about the specific communication protocols used and how communication is established.

[9]Note, in this work, the terms "coordination system" and "EMS" appear always in the same context. We also emphasise that the terms "prosumer" or "energy cells" fall under the same category, because all terms simply describe systems for energy management for a particular environment. In this thesis we use these terms as synonyms.

The hierarchical design with coordination systems introduces several needs for specifications, which we have to consider when we define the behaviour of the systems in addition to the interface specification. The easiest way to explain it is an example, where the coordination system is in an intermediate position. Such a system receives data from its children. After collecting the data, it has multiple options for coordination. The coordination system can directly calculate the coordination signals for its children, to balance the demand and supply, e.g. using an auctioning based model or a short time planning model, depending on the desired coordination design. This behaviour is a classical bi-directional request-reply interaction pattern between the coordination system and its children. It requires to specify the data exchange between two adjacent layers. However, when there is not enough local supply available, the coordination system in a hierarchical structure should be able to ask its parent, whether there is a better option available. In this case, the coordination system needs to aggregate the data of *n*-children and send a request to its parent, who answers the request, after some time. The answer considers only the aggregated data of *n*-children. The coordination system needs to process this signal. It needs to decompose it, so that it can be send back to the children. These two interaction steps require the specification the data processing across multiple layers from bottom to top and from top to bottom. It means they need to define how data can be composed and how it can be decomposed. We deepen the discussion about the composition and decomposition in the next chapters and show concrete examples in the upcoming case studies, since it is easier to explain the details behind that with properly introduced examples.

Further, such a hierarchy affects the interface design. The interface of a child looks always the same for the parent. If an intermediate coordination system receives some data, for instance `MarketBids`, it also sends `MarketBids` to its parent, until at some stage a market clearing process takes place. Another example can be offered flexibilities, which are send to a parent system and composed to flexibilities with more volume. These higher volume flexibilities are again offered to the next higher level. This similarity concerns compositions and decompositions, which are central to hierarchical systems. We discuss those challenges in more detail in the upcoming chapters.

4 | Designing Hierarchical Architectures for SES

In this chapter, we describe the fundamentals to design generic hierarchic architectures in order to realize coordination activities for EMS according the energy system models that we introduced in the previous chapter. The intended architecture shall support the design and integration of systems that use the functionality from child components in order to achieve common system goals. The integration is particularly promising for SES, where most of the components are part of an EMS that offers a set of functionality to balance the energy demand and supply. While there are many special terms for that integration idea, e.g. holons or cellular systems, we use only the term *generic hierarchic architecture* to highlight the fact that we should not be tempted by some extravagant terms to design such a system, but rather focus on available common software engineering best practises.

This chapter starts with a context for the derivation of the generic architecture. It describes the envisioned scope, typical system requirements and the approach for the development of the generic hierarchic architecture. The derivation of the architecture is carried out in the second section. We use well known architectural design patterns to define the basics for our architecture. We explain those patterns in detail and use them to create a generic architecture. After these basic foundation has been laid out, the third section describes the behaviour of the envisioned EMS components. We use basic behaviour blocks to construct the generic behaviour and show their interrelation. To support the technical implementation of the described concept, we introduce a framework that has been developed during this thesis to prepare experimental cases studies. Finally, we give a short summary to wrap-up the most important concepts of that chapter.

4.1 Context

The ascent of new renewable energy systems that enable less carbon emissions, e.g. photovoltaics, wind and biogas systems that are complemented with storage systems, heat pumps or electric vehicles, challenges the current energy system design. The reason is that many of the renewable systems are connected to the lower voltage levels and widely spread throughout the network, while conventional large power plants were comparatively less in terms of numbers and easier to monitor and coordinate. This leads to many questions for integration and automation into an overall infrastructure as pointed out by Grijalva et al. (2011). Many components of the envisioned system address similar functionality, particularly, for the operation and management of energy equipment. Similar functionality opens the possibility to reuse available design concepts, models, control algorithms, etc., but we need to identify the commonalities in the SES domain, particularly from the perspectives of power systems and software engineering. This allows to question, which similar functionalities within EMS are available, how can they help us to describe an architecture, which constraints are imposed on the architecture and which properties are given if we build a system that implements those constraints? In the following we try to approach these questions by understanding the scope and the system environment to identify the most relevant requirements to build such systems. After this foundation, we describe a methodology to satisfy the requirements by well known architecture design patterns.

4.1.1 Scope

Smart energy systems address a wide range of fields and applications. The broad efforts of national and international standardization activities, such as IEEE P2030 (IEEE, 2011), SGAM (ESO, 2012) or CIM (CIM, 2015) (see also Chapter 2), show the immense abundance of required specifications. Each activity cover dozens of components. For instance - only CIM addresses several hundreds of models[1] that cover the standards IEC 61968, IEC61970 and IEC 62325. They are used for the development of SES use cases, interface and behaviour specifications. Further projects related to domain analysis (Irlbeck and Koutsoumpas, 2015; Bytschkow and Ascher, 2017) reveal the large amount of different systems that are used in the context of SES, including SCADA, ERP (Enterprise Ressource Planing), WFM (Work Force Management), GIS (Geographic Information Systems), EDM (Energy Data Management), power system state estimations, power system simulations, power quality measurement systems, predictions, smart meter management and many other specialized systems of system vendors depending on national requirements and market mechanisms. Obviously, not all systems and their features are supposed to be covered by our architecture.

The focus of this thesis is the development of a generic architecture that enables the integration of several EMS into different hierarchical levels in order to provide additional capabilities for coordination and optimization of systems. The architecture aims to expose available internal flexibilities of EMS, which are used in coordination or aggregation activities to offer the flexibility to higher hierarchic levels. The step backwards, i.e. the decomposition of flexibility to smaller fragments and its delegation to the lower levels is the respective step in the other direction. We understand our architecture as a support to establish certain coordination activities that are envisioned by the SES vision. In this vision a hierarchical structure of components is not necessarily fixed. We assume that a component be part in different hierarchical groups, in particular in such with orthogonal operation goals. For instance one component can offer energy market products at two different markets (e.g. day-ahead or ancillary services as introduced in Chapter 3). Here, each market represents a dedicated group with its own hierarchy. Or it can offer a certain percentage of storage capacity to one group and certain percentage to another group. Hence, the envisioned architecture is not fixed on certain dedicated energy use cases, but rather on some generic constraints to define a system boundary that allows a better integration of EMS.

Architectures for interactive EMS involve multiple design decisions. Some decisions address questions for the internal system functionality. These affect internal software components and their composition to cover requirements, such as the internal control of the system or user interaction. Other decisions address questions related to the external integration. These affect externally visible interfaces that enable an interaction with these systems. Particularly the latter perspective determines the integration capabilities of EMS into an overall system architecture, which is the major goal of this thesis. We focus therefore more on the external capabilities in the following. The internal perspective of an EMS is nevertheless still important. We present it in a complement to this thesis in Appendix A. In our complement we demonstrate that the hierarchical structure applies also for internal components and can therefore be used for the internal and external perspectives of an EMS architecture.

[1]We refer to the UML classes defined from the CIM user group driven by the European committee IEC-TC57.

4.1.2 Domain Requirements

The definition of architectures can never be stand-alone. It is always coupled to the requirements that are expected from the desired systems. Taylor et al. (2009) even argues that the requirements and the definition of the architecture are naturally and properly pursued cooperatively and contemporaneously, because our understanding of what works now, and how it works, affects our wants and perceived needs. The starting point to create a new architecture, thus, is based on the knowledge about the systems that exist now, their architectures and requirements about all that is desired that current system fail or fail to provide. We introduce our starting point in the following, by stating several important requirements for our system architecture.

Distributed Energy Management Systems

EMS are geographically widely dispersed at various places and facilities. From large enterprises to small companies and private households, all systems should have the possibility to participate in market oriented SES applications, applications for network control, or both. Thus, there are two requirements. First, the architecture shall consider that systems are located at **remote locations**. Second, an EMS consists of components that support autonomous decisions based on their environment and their internal or external stimuli. Hence, EMS can receive or publish data and perform some actions leading to **proactive and reactive** interactions.

Further, most systems are created with the assumption that they are governed by a single entity. This is different for collaborating EMS systems. They originate from multiple organisations and different vendors with multiple organisational boundaries. This implies that deployment and commissioning are not necessarily synchronized. New components need to be gradually integrated into the system while slowly exchanging the old ones. The architecture needs to ease such **independent deployment** and allows that components co-exist at different life time cycles.

Closely related to the independent deployment is the requirement for **extensibility**. Even if a system perfectly matches current requirements, regulators, developers or users will always have new ideas and desires, which need to be integrated. To enable a good extensibility, clear interfaces and APIs are key specification areas for crucial for good interoperability.

Multiple organisations also imply that multiple levels of trust exist. Security becomes a significant concern. Strong security measures should crucial, since the participation of untrusted users and applications demand for authentication, authorization, and control of exchanged data. The requirement is that state-of-the art **security technologies shall be supported** by the architecture. Security is stated here to highlight its importance. However, suggestions to achieve this are beyond the scope of this work.

Further, SES are expected to cover thousands of components. Thus, **scalability** is another crucial requirement. Scalability means that systems are able to grow without strong limitations from single components. Sometimes, it implies that components cannot know all participants of a system, the complete system state or be responsible for the operation of the whole system.

Note, similar requirements also appear in development of web applications. They are well explained by Fielding (2000), who presents a web architecture reference that is also known as Representational State Transfer (REST) architecture style. In his work he demonstrates how requirements are addressed with best practices and architectural design patterns as well.

Automation Systems and Physical Interdependency

Much of the recent development is targeted towards the automation of energy systems. Particularly systems for (real time) monitoring and control are of interest. Such systems today are responsible to collect data in the field, (pre-) process it locally, send it to a monitoring system and persist it for documentation and (statistic) analysis. To enable the **integration of automation systems** we need to understand them in terms of their architecture and find suitable solutions for integration. The architecture of automation system follows a layered, pyramid-shaped structure as shown in Figure 4.1 (left). The pyramid shape is used to emphasize high data flow frequencies between the low-level components, and lower frequency at the higher-level (Vogel-Heuser et al., 2009). To enable their integration, security and safety reasons demand that the hardware used for control is not exposed. In the best case, the available hardware and many other internal details remain hidden. This is known as **encapsulation**. Thus, we require to define interfaces on the higher levels of the architecture and restrict the internal access to the lower levels.

Interfaces to higher level systems are today discussed frequently, especially due to the rising importance of IT business systems that demand for an intensified cross-linking between the levels. This led to a rethinking of the pyramid shape. To highlight this intensified cross-linking a double truncated cone with an information model as the intermediate layer (see Figure 4.1 right) has been introduced in general by Vogel-Heuser et al. (2009) and for particular execution system applications by Keddis (2015). It intensifies the connection between three layers: the business logic layer on top, the intermediate information layer in the centre, and the process layer at the bottom. It also means that operational goals or settings for the process and field layer are not necessarily defined on the control layer, but also systems from higher levels might be responsible for the automation of SES. The interoperability needs therefore **flexible and extensive information models** that can be combined with the process level in automation.

Finally, since EMS have a strong physical interdependence the execution of control signals has an impact on the power supply. Thus we require a **consistency of the expected behaviour**. It means that we need to define an expectation how certain signals are translated into physical actions that are for instance part of a mathematical model for coordination (e.g. short term planing).

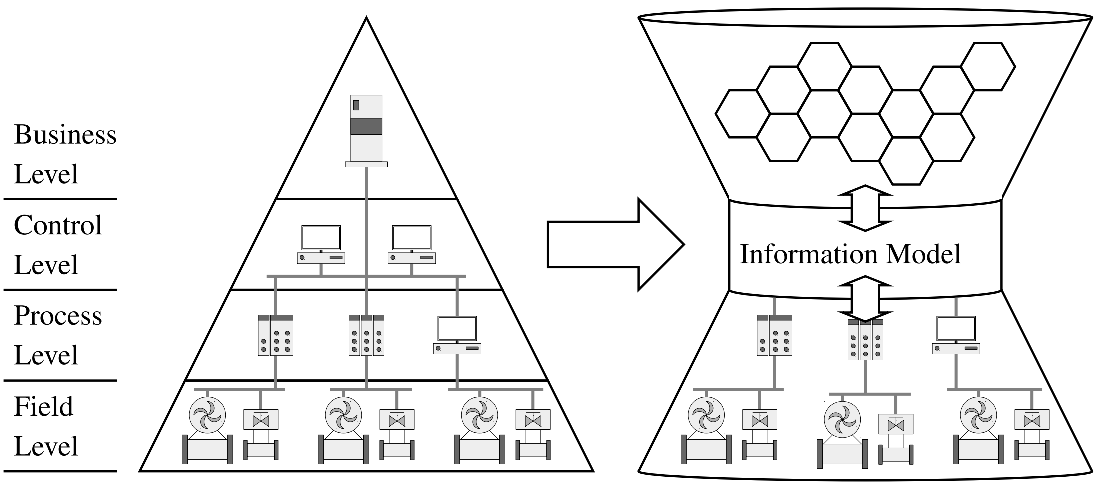


Figure 4.1: The illustrated shift of architectures in automation and manufacturing systems (Own illustration based on Vogel-Heuser et al. (2009)).

Geographical and Operational Scopes

Systems of the smart energy domain experience recurrent hierarchical patterns in their geographical and operational scopes. Geographical scopes denote the liability for an energy supply infrastructure of a defined area. We observe EMS that are responsible for single entities or a small local area, DSO[2] systems that are responsible for distribution in cities or rural areas and TSO[3] systems that are responsible for the transmission infrastructure that covers large distances. Each authority is responsible to observe and clear all events that occur in their responsibility area. Their hierarchical structure act as an **aggregation** mechanism. Geographical scopes have physical connections to adjacent hierarchical levels as well as to neighbours at the same level. Such transfer points (e.g. power substations) define the boundaries of the geographical scopes.

Operational scopes define additional system boundaries of issues that are not necessarily related to grid operation. These are manifold, e.g. trading of power at the market, integration of facilities into virtual power plants, and all kind of auxiliary services related to the energy domain. Operational scopes define **aggregations of components** to provide better services, as scaling effects lead to higher efficiencies and better short term forecasts. Operational scopes can also change from time to time, since regulation requires a free choice for customers to select their service providers.

Software systems are important for both scopes, geographical and operational. One requirement is therefore that **different hierarchic affiliations** are considered in our architecture. Tolerance of different hierarchic affiliations is a key requirement to handle situations, where we have more than one superior hierarchy system. Contrary instructions of hierarchic affiliations can be avoided by orthogonal responsibilities, which do not interfere on the same resources and system's state. Another possibility to tolerate multiple hierarchies are clearly separated interfaces. This is in fact a good practice for software engineering since it facilitates the separation of concerns leading to more flexible systems and establishing loose coupling between components, which can be started, stopped, updated and maintained independently.

We summarize the requirements in Table 4.1 for a better overview and as reference to map them to the architecture design patterns later on.

Limitations

We presented a set of requirements as a necessary prerequisite to describe what we want to achieve with our architecture. Of course any list of requirements can be refined and extended by further specific requirements that are necessary for particular SES applications. Actually, there are many good literature surveys that specifically collect SES requirements, for instance by Rohjans et al. (2012). Our selection of the desired requirements focuses on the integration of EMS on the general level and is not application specific, since we want to specifically tackle the architectural question of building a system that support the hierarchic coordination of EMS. With our approach we target to achieve a higher abstraction so that we are not necessarily limited by single applications and can potentially generalize our architecture to several use cases later on.

[2]Distribution System Operator

[3]Transmission System Operator

Requirement	**Description**
R1: Remote locations	The generic architecture shall integrate remote EMS.
R2: Proactive and reactive behaviour	The generic architecture shall integrate systems with proactive behaviour, which triggers some process, as well as with reactive behaviour, which reacts on triggers and responses appropriately.
R3: Independent deployment	The architecture shall enable that the systems can be to deployed, stopped and started independently.
R4: Extensibility	The architecture shall support system extensions. It means that a system might be refined and extended, but due the distributed nature, it cannot be guaranteed that every system is extended synchronously. Therefore, non-synchronous refinement capabilities are required.
R5: Scalability	A large number of components shall be supported. One single component shall never limit the total system.
R6: Support of security mechanisms	State-of-the-art security mechanisms shall be compatible with the architecture.
R7: Integration of automation systems	Automation systems shall be integrable in such way that the architecture does not require the direct access to hardware components, since encapsulation is a major mechanism to protect automation systems.
R8: Flexible information models	Coordination of EMS involves that control systems are interconnected using similar technologies as business software systems. The architecture shall support those techniques and allow to design flexible information models to adapt the interface to different coordination approaches.
R9: Consistency of expected behaviour	EMS systems shall provide a clearly defined behaviour to allow its usage for the control of physical systems.
R10: Aggregation of components	EMS systems shall be able to aggregate components. Such aggregations represent groups of components that act as one single component to the upper hierarchy levels.
R11: Different hierarchic affiliations	EMS systems should be able to take part in different hierarchic affiliations.

Table 4.1: Overview of the requirements for the hierarchic SES architecture.

4.1.3 Approach to Derive the Generic Architecture

Deriving architectures for interoperability and specification of interfaces is often the main objective of standardisation activities, e.g. IEEE (2011); ESO (2012); CIM (2015). Much time and effort is spend on good specifications and methods to derive them. There are many different approaches to derive architectures as pointed out by Irlbeck et al. (2013), for instance top-down approaches as the IEEE P2030, approaches that provide only certain guidelines with predefined structure as the SGAM methodology or bottom-up derivations as carried out in the extensive work by Irlbeck and Koutsoumpas (2015). All those approaches propose reference architectures that guide systems architects and developers with common terminology to improve knowledge transfer, enable system comparisons and introduce concepts like software components that can be reused to facilitate design and integration of new components. Our work is complementary to the broad domain analysis, since we have done some previous work in that area, for instance in Irlbeck et al. (2013), Bytschkow and Ascher (2017), Ascher and Bytschkow (2018). But our major contribution here is to derive an architecture that uses energy system models for coordination as a base line to allow the integration of different EMS and demonstrate how the system benefits from this integration. We propose an approach that is based on best software engineering practises that are known from other domains and combines them with energy related modelling techniques to achieve the desired coordination of EMS.

The generic architecture is created from a set of architecture design patterns that induce a set of desired properties (see also Chapter 2). It can always be enriched by application specific demands and extended by specifications to meet the domain requirements. The idea can be described more precisely. We suppose that well known architecture design patterns exist. They describe roles and responsibilities of components in a system. Hence, they provide a set of clear principles to create a system architecture. An architecture design pattern addresses a set of requirements and it has certain benefits and weaknesses (Fielding, 2000; Taylor et al., 2009). For instance, a client-server pattern describes two classes of components that communicate over a network. A server is a component that offers a set of services and listens for requests upon the services. It is a reactive process. A client sends requests to the server to trigger an execution of services. It is a triggering process. The client-server pattern is used for a clear separation of concerns in a system, but it does not constrain the partitioning of functionality. For instance, we have systems with 'thin' and 'thick' clients, server applications and particular backend systems. The client-server pattern is well described, e.g. by Andrews (1991) or Taylor et al. (2009).

Our derivation of follows a constructive stepwise approach, similarly as for the development of REST (Fielding, 2000), the underlying architecture design pattern for many web-applications. A good practise is to start with a NULL style, where no principles that constrain the system are present. New patterns are added step by step by explaining their contributions. This allows to check every step for inconsistencies with the previous set of design patterns and by that analyse if the envisioned impact is reached step by step. Whenever conflicts become obvious, the modification can be adapted or weakened to emphasise prioritised properties. It is possible to add well known and industry proven architecture design patterns, that have been used in many real life projects as basic building blocks, and combine them to reach a new hybrid architecture design pattern. Since well known architecture design patterns are known by many system architects, a combination of two patterns can be evaluated for potential conflicts.

4.2 Derivation of the Generic Architecture

In the following, we derive the hierarchic SES architecture step by step. We begin the derivation of the generic architecture with an empty set of constraints. Then we add step by step design decisions, i.e. well-known architectural design patterns, by describing concretely what exactly is necessary using our list of requirements (i.e. Table 4.1) and how each individual architecture design pattern helps us to achieve those requirements. Note, the intention of the architecture is neither to restrict the communication of components to a particular protocol, nor to propose certain technologies for the interface, but rather to provide a clear but technology independent specification and describe the expected interactions between individual components.

4.2.1 Required Architecture Design Patterns

Client-Server

In a system with *remote locations (R1)* and communication between components we start with the addition of the client-server design pattern as the first set of principles to our architecture. The client-server pattern is based on well known and clear responsibilities. As we shortly described before (Section 4.1.3), the pattern introduces the two roles clients and servers with certain interaction expectations. Servers wait for request messages. Clients initialize the communication and ask for services from the servers. The server responses with answers that depend on the service. Note, if the service is not available there are different protocols that provide certain status codes and support the communication, for instance the HTTP-protocol. These codes are however an extension to the server-client pattern. The pattern it-self restricts just the roles and constrains that clients do not interact. The pattern is depicted in Figure 4.2 to illustrate the idea.

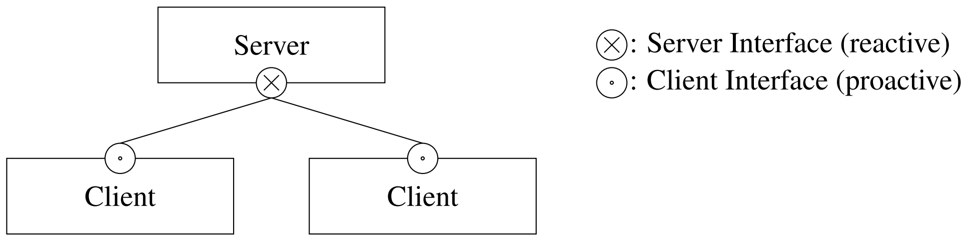

Server
Client
Client
⊗: Server Interface (reactive)
⊙: Client Interface (proactive)

Figure 4.2: Client-server architecture design pattern.

A rich set of libraries and available systems is available to realize a client-server architecture. Besides the requirement of remote locations, the client-server pattern allows to consider *(R2)* that we expect for SES: *reactive behaviour* (server) and *proactive behaviour* (client). The second requirement is therefore covered as well. Probably the most significant contribution of the client-server design pattern is that it is specifically designed for a distributed environment, where systems can be started and stopped independently. It is therefore perfectly suited to realize an *independent deployment (R3)* of different systems and multiple organizational boundaries. Even if some systems have been stopped and are not reachable for some time, the non-monolithic architecture enforces that all the other components can still operate and do not necessarily break down the whole system.

Composite Pattern (self-similarity)

Next, we would like to add a restriction for our architecture that affects the interface design and available services of the components. For SES we expect that EMS like prosumers or microgrids offer a similar set of services, which are repetitively found at every hierarchic level. The idea is that, no matter on which hierarchical level the system is available and no matter whether the system is an individual component or a composite of several components, the system should be treated (integrated) uniformly or in other words transparently. The restriction of a uniform interface demands that there is common set of services that are provided over the interface of each EMS [4].

A well known mechanism that enforces unified treatment is the composite pattern (Gamma et al., 1995). Originally it is a structural pattern for the composition of objects in object-oriented programming that provides a hierarchical composition of components that have the same set of services. It is successfully used in many real application that demand a generic approach and good *scalability*. File systems or graphical user interface toolkits, such as Java Standard Widget Toolkit (Java SWT), are good examples. In our architecture, we use the term composition pattern to denote a structure for the composition of software components that interact via interfaces and exchange messages. The composite pattern restricts therefore the architecture to a hierarchy of components that offer a defined set of services and consequently have similar interfaces, even though each component is responsible for its specific hierarchical level.

The composition pattern restricts the system architecture to two different component types. The first type is a leaf component. It represents an atomic component that cannot be broke down further. It has the defined set of services, but no children. The second one is the composite component. It offers the same set of services and in addition to leafs, it can add or remove other components and by that cover different system compositions. The composite pattern adapted to our component based architecture is illustrated in Figure 4.3.

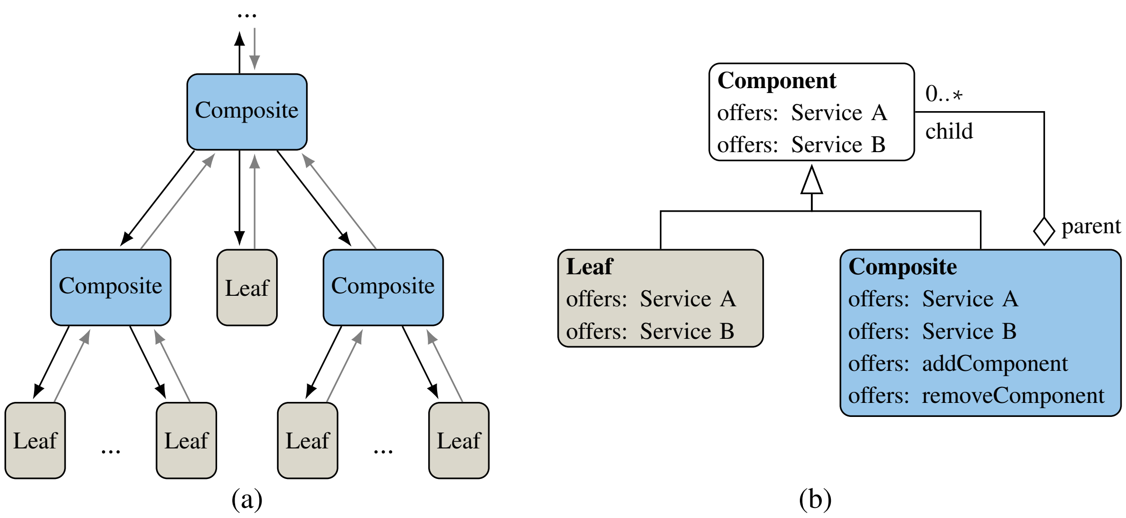


Figure 4.3: The composite pattern: (a) hierarchic system architecture of interacting of components, (b) components as composites or leafs.

[4] Note, of course each individual EMS still might have other services (vendor specific ones), but to reach the possibility to coordinate EMS using some common model, each component should be addressed uniformly.

Combining the composite pattern with the client-server style introduces the restriction that each component acts potentially as a client and a server. The important contribution of the combination is that composites and leafs have the same interface and behaviour. We therefore say that they are self-similar.

The composite pattern is useful for our generic hierarchic architecture to address three requirements. An architecture that is based on the composite pattern supports the composition and decomposition of functionality. It allows to delegate requests from parents to children, to aggregate information from the children and offer a better service to the parent. Consequently, the *aggregation (R10)* requirement is addressed by the composite pattern by the design, but due to its generic structure the exact definition of the interface depends on the application, which we discuss in more detail in Chapter 5 and Chapter 6.

Further, most of the information processing and coordination can be achieved through the intermediate composite components that represent groups of components. This reduces the necessity to send all information all the way to the top level component, which increases *scalability (R5)* of the system. Scalability is further supported by the possibility to cache data at intermediate components. In addition, whenever new components appear, they can easily be hooked up to any composite components, without influencing the topology of other systems. This provides a good way to share the work load of services. The scalability property can decay when we do not consider two factors. On the one hand, a single composite should handle only a limited number of children, especially if it designed to aggregate the children's results and the aggregation process is complex (e.g. some kind of optimisation). The limit should be chosen such that the composite is well prepared to operate within its performance requirements. The other factor that might decrease scalability is the depth of the hierarchy, especially when the communication processes are designed in such way, that they require many request-response interactions that are passed to next hierarchy levels. In the SES vision each EMS (every hierarchic level) covers its responsibility as autonomic as possible. Request-response interactions over several levels should therefore be reduced to a minimum.

The third requirement that is covered is *consistency of expected behaviour (R9)*. Every component of the composite pattern has a similar interface with same functionality (cf. section 4.2.3). Actually, the biggest advantage of the composite pattern, is that it allows an agent to interact with any composite component (which represents a group of components) exactly the same way as with leafs. This allows to specify an interaction only once for the composite components. The other components should follow the same specifications. The combination of the client-server style with the composite pattern is depicted in Figure 4.4.

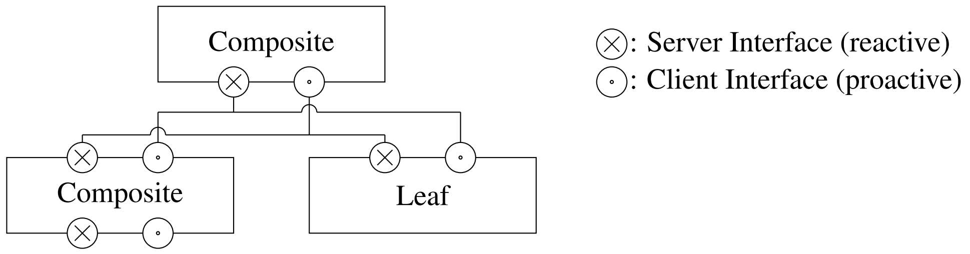

Composite
Composite
Leaf
⊗: Server Interface (reactive)
⊙: Client Interface (proactive)

Figure 4.4: Client-Server + Composite Pattern.

The combination of the composite pattern and the client-server approach already provides a good set of constraints to realize many applications. Those two principles are a good starting point to specify the design and behaviour of EMS component. Depending on the desired coordination approach, we can choose which functionality is provided as a server and which as a client, e.g. flexibility offers or market bids. We can also reason about those systems, which initializes the interaction and which information is transferred. This is demonstrated in Chapters 5 and 6 that focus on specific coordination approaches.

Layered Systems

The next constraint of our architecture is the layered architecture design pattern. Layers are well known from communication systems. They are used to organize a system's functionality hierarchically. Each layer provides a set of services. The services use other services of the layer below it and provides their functionality to next adjacent higher layer. By hiding the inner layers from the outer ones, the system's complexity is greatly reduced. In strict layered systems skipping of layers is not allowed. This increases re-usability and evolvability of systems, as layers might be exchanged or adapted individually. Layered systems are implemented in many real word applications. Most communication systems, such as the TCP/IP (Fall and Stevens, 2011) follow the OSI model (Zimmermann, 1980) and use layered protocol stacks. Further examples are operation systems with their hardware drivers and abstraction libraries. Systems in the energy domain with gateways and proxies or SCADA systems widely implement layered system mechanisms to protect their internal functionality from undesired access.

The principle of layered systems helps with two further requirements. We required that our architecture does need to *support state-of-the-art security mechanisms (R6)*. A layered system allows to add additional layers for EMS, e.g. smart meter gateways, and add security mechanisms such as VPN, SSL/TLS to interact with the systems. In fact, there is already a lively discussion by the regulation bodies with a specification according the Common Criteria by the Federal Office for Information Security[5], about which requirements such a smart meter gateway must fulfil and how it will be evaluated to obtain a certification in Germany. A layered approach is part of this specification.

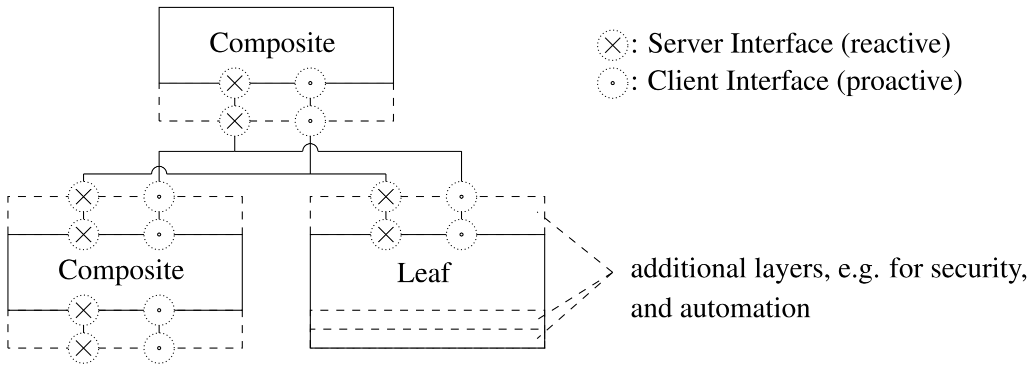

Composite
Server Interface (reactive)
Client Interface (proactive)
Composite
Leaf
additional layers, e.g. for security, and automation

Figure 4.5: Client-Server + Composite Pattern + Layered Systems.

[5]Bundesamt für Sicherheit in der Informationstechnik

The second requirement that is covered by layered systems is the possibility to *integrate automation systems (R7)*, that the only one particular layer of automation system is accessible. This allows access to internal components to be reliably hidden to enforce encapsulation requirements. We introduced in Section 4.1.2 that automation in general and SCADA for energy systems in particular are developed with a layered approach in mind. Even the previously mentioned Common Criteria specifications discus the question, how to open the possibility to address automation scenarios with the smart meter gateways. The principles of layers are therefore of paramount importance for a generic architecture in the energy domain. The addition of layers to our architecture is depicted in Figure 4.5. In Appendix A we also provide more details how layers are used in SES prosumer systems.

Unified Interface

Our last three principles already describe the most important structural constraints for our architecture. We add the last obligatory architecture restriction, that we denote a unified interface, to emphasise additional requirements to create interfaces, which are easy to integrate and not bounded to specific technical protocols. With the term unified interface we postulate certain expectations about a system's interface [6].

A unified interface demands that external available system capabilities are represented as addressable resources. Any resource can be accessed by a unique identifier (an address). Any resource is accessible by a representation that chooses a certain protocol and data format (e.g. JSON, XML, CSV, or some other format) and multiple representations potentially exist in parallel (it means that the system might provide its resources with different representations and also technical protocols (e.g. HTTPS, OPC UA), but the unified interfaces focuses on the definition of a resource, not on its technical implementation). The current state of a system is a resource as well. Finally, in a unified interface available options for further interaction are explicitly shown. It means that resources, which require other resources, refer to each other.

The unified interface principle addresses several mentioned requirements. First, the focus on addressable resources allows that *different hierarchic affiliations (R11)* use and interact with differently available resources, since the resources can be designed so that each affiliation receives its own set of resources.

Further, the *extensibility* requirement *(R4)* is supported, because the system's capabilities are presented as a set of addressable resources. Resources can be added or modified and old ones removed. A resource can have multiple representations, which can be extended if new application requirements or protocols appear. To facilitate systems compatibility, old representations can also be kept active, while new ones are added stepwise. This supports a continuous development, testing and an independent deployment.

[6] Note, the term unified interface also appears in the work by Fielding (2000), where it is introduced as a major ingredient of the REST architecture style, a widely used principle that particularly specifies system interactions. It defines four principles for an interface: identification of resources; manipulation of resources through representations; self-descriptive messages; and hypermedia data as the engine of application state. This definition was fundamental for the work on the world wide web.

The *aggregation* requirement *(R10)* is supported one more time by the explicit representation of the current system's state and its capabilities. This allows to observe the state of every component. State updates can either be pushed upwards the hierarchy, or pulled from the lower levels, depending on the desired implementation, which we do not restrict here. We represent the unified interface in the architecture style (see Figure 4.6) by the symbol U to denote a component, which supports such an interface. An extension with colored blocks for every component represents the different possibilities for the representation of resources.

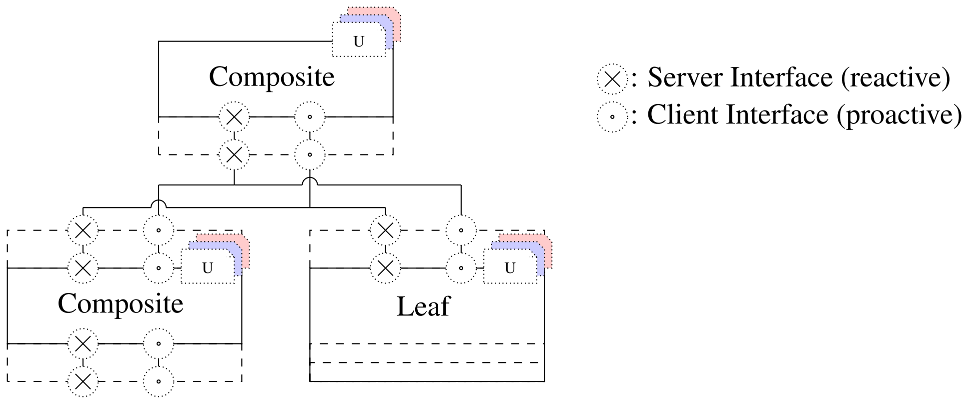


Figure 4.6: Client-Server + Composite Pattern + Layered Systems + Unified Interface.

Publish-Subscribe (optional)

Finally, we add one more optional architecture design pattern to explicitly highlight and support automated, continuous data exchanges in SES to achieve a higher convergency towards the *integration of automation systems (R7)*, namely the well-know publish-subscribe pattern. The publish-subscribe pattern introduces two distinct components: the publisher and the subscriber. The publisher periodically creates information, e.g. new measurements or system states. The subscriber obtains this information or is at least informed that new information is available. This reduces the communication overhead and the necessity to check for new information, as long as no new information is available.

Further, the publish-subscribe design pattern demands that the publisher maintains a list of subscribers. Subscribers can subscribe to that list and also deregister their subscriptions. This allows to establish more flexible connections and adapt the communication channels flexible, if required by the system. The publish-subscribe design pattern offers another benefit in practise. First, with the rise of many new IoT applications, there is a strong technological support from many industrial partners that also offer libraries for the implementation an well standardized communication interfaces, e.g. MQTT, OPC UA. Second, the developers of such systems can focus more on the question which data is required and how it can be processed, instead of solving the technical questions about stable communications and how to handle communication errors, since those questions are usually covered by the already available technologies.

The publish-subscribe pattern is not really mandatory to create the hierarchical applications, as it the architecture can be established also with other approaches, e.g. REST or some bus technologies that have no dedicated requirements for publish-subscribe mechanisms. But the publish-subscribe pattern greatly facilitates the automation capabilities of components. This is why this architecture design pattern is marked as optional. To illustrate the addition of the publish-subscribe pattern to our architecture, we extend the unified interface symbol as ↑U↓ . The resulting combination of architecture design patterns is illustrated in Figure 4.7.

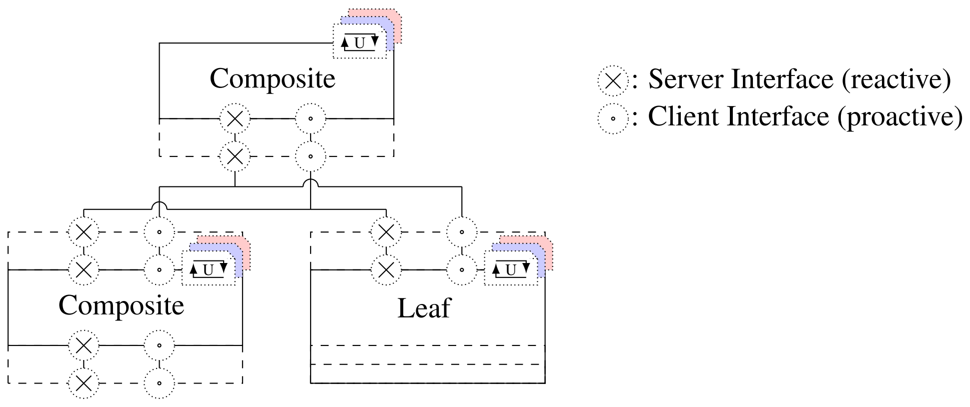

U
Composite
⊗: Server Interface (reactive)
⊙: Client Interface (proactive)
U
Composite
U
Leaf

Figure 4.7: Client-Server + Composite Pattern + Layered Systems + Unified Interface + Publish-Subscribe.

Additional remarks on channels

For the work on the architectures, it is worth to note some practical remarks here. Many architectures introduce components and connectors to model software intensive systems (e.g. Broy and Stølen (2001), Neubeck (2012), Vogelsang (2015), Junker (2016)). The components represent certain functionality (system behaviour) and have an interface (system boundary). Connectors represent data links. Therefore, an architecture often looks like a graph made from blocks (components) and arrows (connectors). This is done to abstract architectures from implementation details and focus only on the details that matter for the specific application. Such separation of concerns is good software engineering practise and very beneficial, as the system functionality can be evaluated or even formally verified with clear specifications and less complex system descriptions.

We should keep in mind, though, that particularly connectors in implementations are often not just pure data links that transmit bits and bytes. In fact, connectors are responsible for many important actions, which are often abstracted away in an architecture model. This includes specific protocol implementations, invocations, caching, persistence, messaging, transactions, routing and sometimes even security related interactions. Consequently, the choice of connectors do greatly affect the properties of systems. In contrast to components, that are usually application specific, connectors are more general and can therefore be reused across applications and domains. In consequence, many re-used libraries in real system implementations are connectors.

4.2.2 Mapping of Architecture Design Patterns to SES Requirements

We described the generic hierarchic architecture as a system that is constructed stepwise from other well-known architectural design patterns that are often applied in real systems. They define the necessary architectural constraints for the generic hierarchic architecture. We use the constructive approach to cover a set of requirements (i.e. Table 4.1). This clarifies the rationale behind each decision and makes it easier to understand the implication of each choice. A summary of the used principles and the addressed requirements is presented in Table 4.2. They focus on the SES context, but are not limited to that area.

Architecture Design Pattern	Addressed Requirements
Client-Server	Remote locations (R1) Proactive and reactive behaviour (R2) Independent deployment (R3)
Composition Pattern (Self-Similarity)	Scalability (R5) Consistency (R9) Aggregation (R10)
Layered Systems	Security (R6) Integration of automation systems (R7)
Unified Interface (Representation of Resources)	Extensibility (R4) Flexible information models (R8) Different hierarchic affiliations (R11)
Publish-Subscribe	Integration of automation systems (R7)

Table 4.2: Overview of the required architecture design patterns and related requirements.

An additional benefit of the stepwise construction is that the used architecture design patterns come along with a set of technologies, such as available libraries, tool support and methods, which help to understand, design, analyse and implement such systems. In the upcoming work we give several examples how these presented principles are translated into real systems, which are used in our real life projects.

4.2.3 Architectural Elements: The Role of an EMS for Coordination

The previous section describes the basic design patterns to create a generic architecture for the coordination of EMS and explains why we selected them. The architecture design patterns introduce only carefully selected constraints that are important to design a properly functioning system. To understand how this creates an architecture, we further need to understand what are the responsibilities of each component in the system as clearly as possible. Hence, we need to define the roles of the involved components. The description of the roles is intended to be technology agnostic and leave the freedom for developers to choose certain technological solutions or communication protocols.

On the most abstract level, a component in our generic architecture is an EMS, for instance a prosumer system, a microgrid, a coordination system for multiple prosumers or microgrids or a coordination system for multiple other smaller coordination systems. All EMS offer some defined set of services that depend on the desired coordination model. With the common set of services, they also provide similar interfaces (in terms of available data) and behaviour (in terms of data processing or in other words input / output relations for the data). We presented that an EMS can be a leaf or a composite. But, since all EMS are just a specialized form of the general EMS component, the EMS role is only component role that requires a specification.

An EMS is an interactive component in a hierarchical system, that continuously offers a set of resources and requests other resources to perform certain tasks. An instantiated EMS is always either a **composite** or a **leaf**. If it is a composite, the EMS controls and continuously collects values from its children. If it is a leaf, the EMS controls and collects values from its underlying local systems. In the SES context, these are physical devices such as simple sensors, advanced controllers, or even complex equipment (e.g. solar panels, inverters, batteries, heat pumps, etc.). The collected data enables the EMS to compute the available resources.

In a hierarchic structure (see Figure 4.8) EMS interact by sending messages to the higher level components (parents) or to lower level components (children) and also respond to such. The hierarchical interplay and the exchange of messages opens a wide range of possibilities to define different behaviour patterns depending on the desired application. Hence, the interaction in a hierarchical system might be complex. To reduce the complexity we break down the possibilities for message exchanges into atomic steps. This allows a deliberate behaviour specification of a component.

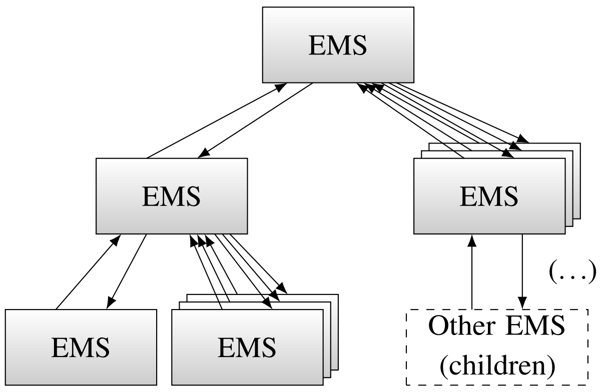
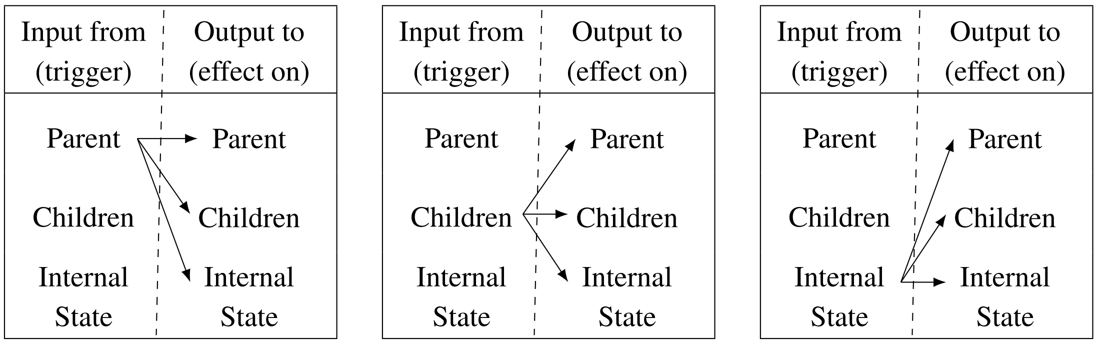


Figure 4.8: The structure in a generic hierarchical architecture with EMS.

For the decomposition of the potential interactions into atomic steps we use a distinction of cases, since there are only a limited number of possible input and output relations. A convenient distinction can be made by the triggering mechanism (input messages that triggers some behaviour) and the sequential reaction of the component. We display all trigger possibilities and potential continuations that an EMS can have in Figure 4.9. A trigger might originate from the parent EMS, from the child EMS or from internal states of an EMS application[7]. Each trigger has a number of possible successive options.

[7]Note, an internal state represents internal system triggers, for instance a human user that desires some action, or from external applications such as smart home applications.



(a) Trigger: Parent's msg. (b) Trigger: Child's msg. (c) Trigger: Internal state

Figure 4.9: A distinction of cases for the specification of the EMS behaviour.

The reactions are the expected behaviour of the EMS component in a hierarchic architecture. We use the reactions as atomic steps in the following to define the behaviour within our architecture. Since we are all familiar with a conventional organisation hierarchy, the specification of an EMS is rather intuitive by following the hierarchical arrangement. The lowest level components, the leafs, provide resources for an operation. They control and monitor physical devices and abstract from internal details, such as implementation of device specific protocols. Composite components have the responsibility to aggregate and manage the available resources of their children in the higher level. Composites provide goals and support with additional information.

An example in our context is the planning and provision of power. The overall goal is to reach a balance with power production and demand. Individual EMS systems are not aware of the global situation, but they can control (or at least monitor) their own system. The monitoring systems provide data about local power injections[8]. These measurements are used to enable the planning for the upcoming future. Parents can collect such information from their children and trigger a planning process. This process results in some coordination signal for the children. Such signals are for instance set points for a desired power injection for the upcoming time periods, or prizes that guide the children and motivate them to shift their demand to lower price time intervals, or utilize more of the available power production during high price time intervals. Children are responsible to react on these signals. If they are leafs, they control their physical devices. If they are composites, they decompose the signal into individual signals for their children and send those downwards in a top-down planning process. Alternatively, planning can be organized in a bottom-up approach. Children send their desired power injection to their parents. Parents monitor the contributions and adjust the incentives to reach the desired goal. In this bottom up process, children trigger the planning process and provide the forecasts. The parents are still responsible for coordination and can use any kind of incentives to guide the children in order reach overall system goals, for instance providing economic power supply under the constraint of system stability.

[8]The term power injection means that power can either be taken from the network (power demand) or provided to the network (power production).

As presented in Figure 4.9, EMS in hierarchical architectures encounter a number of possible interactions. We go through them in more detail in the following to differentiate between the possibilities. We always consider one input/output relation as one atomic case or in other words one atomic behaviour, similarly to the request and response pattern of servers and clients. Due to hierarchy, we say that the receiving of input is the first step, thus it is a *trigger*. The sending of an output is the second step, thus it is a *reaction*. In the following, we present the atomic behaviour cases that require attention for the behaviour specification of an EMS. To demonstrate their application in more complex scenarios, we provide later on two industrial case studies in Chapter 5 and Chapter 6, where we demonstrate implementations of several hierarchic applications for the coordination of EMS.

Distinction of Cases (Behaviour Specifications)

Trigger: Parent's messages.

The first case distinction is based on messages from the parent. We depict all possibilities in Figure 4.10. **The EMS component for which we specify the behaviour is pictured as a shaded circle.** Its parent and its children are denoted with P and C, respectively. The trigger (the input of the behaviour) is denoted as arrow #1 and the reaction (the output of the behaviour) is denoted as arrow #2.

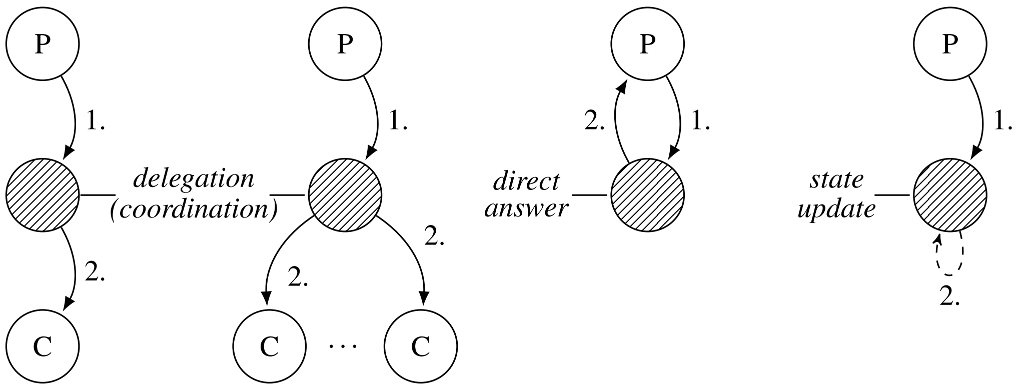


Figure 4.10: The specification of the EMS behaviour based on parent's input.

If an EMS receives a message from the parent it has four options. An EMS can *delegate* the request either to a single child or it can split the request in several sub-requests and delegate them to several children. A delegation behaviour should be specified whenever a request concerns the collaboration of multiple components, e.g. different systems at various locations, but also if it requires a resource that is offered by a specific child. The delegation behaviour can be seen as a coordination activity, where a group tries to reach a common goal. *Delegation* is only possible for composites.

The next possibility for an EMS behaviour is a *direct answer*. This behaviour allows to specify a direct client-server interaction, where the EMS is a server and the parent can contact it directly, thus the parent is a client. The specification of this behaviour is useful, when a component is able to respond directly. For instance, the parent can request an update of the current state. *Direct* *answer* behaviour specifications are of interests for both EMS types, composites and leafs. The fourth possibility for an interaction is a request that leads to a *state update*. Such an example is a request to change a set point for some internally´controlled devices. The set point changes the internal state. The *state update* specification is important for both instances, composites and leafs.

It is of course possible and even preferable, to combine multiple atomic behaviour pattern in a hierarchic EMS coordination to realize more advanced applications, since the the simple steps are a strong limitation. But the atomic steps are still a good guideline for behaviour specifications. For instance, it is reasonable to combine *delegation* and *direct answers*. This allows to inform the parent that its request is being processed, while the children do the actual work. The combination of an *state update* and *direct answer* is also useful to acknowledge that set points were received and successfully established. The exact specification of the behaviour, however, depends on the application requirements.

Trigger: Child's and children's messages

The second triggering mechanism is a message from the children level. Consequently, it applies only for composites, not for leafs. We differentiate in the following between a single child and a group of children to provide more possibilities to specify cases. We depict the different options in Figure 4.11 and structure the atomic behaviours based on the received input (arrow #1) and provided output (arrow #2).

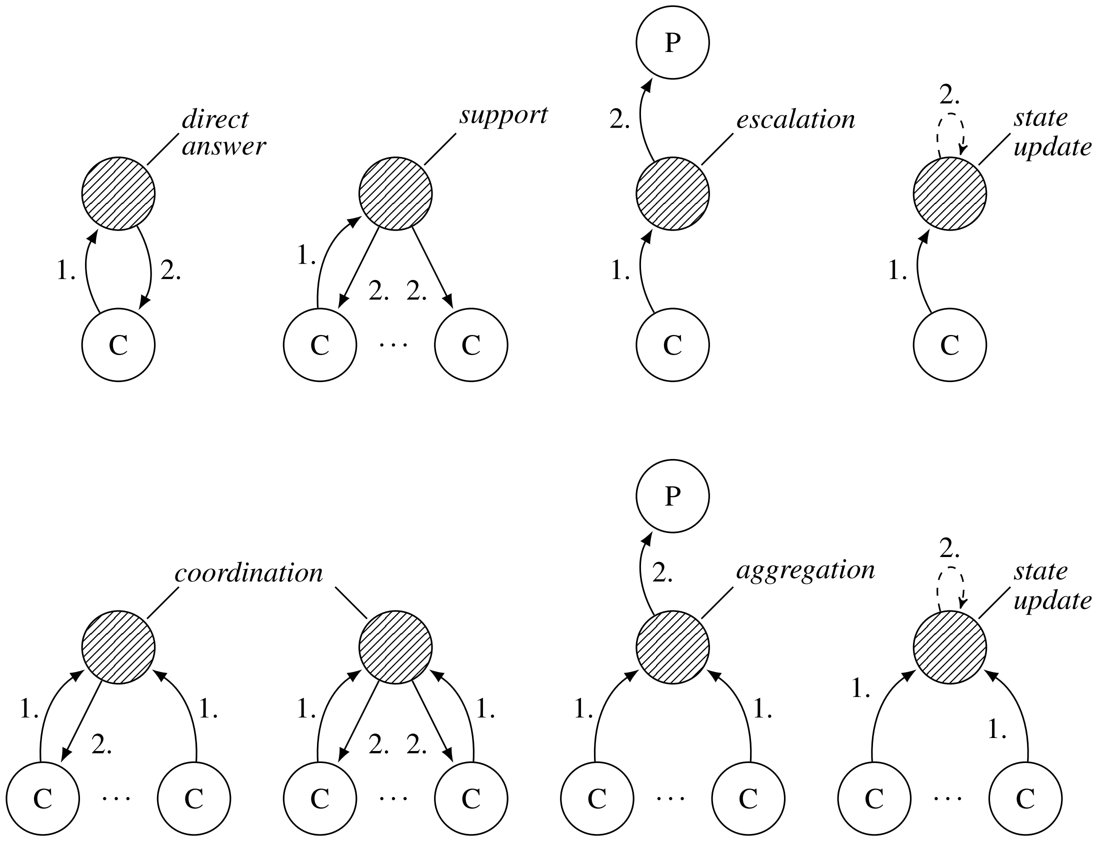


Figure 4.11: The specification of the EMS behaviour based on children's input.

The upper set of cases in Figure 4.11 describes the behaviour based on inputs from a single child. The simplest two possibilities are the specification of a *direct answer* and *status update*. This is similar as explained before. The EMS system receives a request and responses with an answer, or changes its internal state, accordingly.

Another important case triggered by a single child is called *support*. For instance, an EMS child schedules its internal resources. It might appear that it has not enough resources, for instance due to deviations from the forecast. In this case it can send a message to its parent to request support. The parent might know from previous interactions that is can request additional resources from other children. In this case it can activate the resources from those other children to keep the group's total demand and supply balanced.

The final case of the upper set addresses *escalation*. A child sends a message to the EMS. But the EMS does not respond immediately, since it requires support from the parent level. For example an alarm can be escalated, if it cannot be resolved internally, but there is a possibility to receive support from the higher level. Actually, in a highly automated system, it is expected to resolve the incoming tasks on the lowest possible level. Only if this is not possible *escalation* is required. Reasons for escalation can be manifold and application specific.

The lower set of cases in Figure 4.11 describes the trigger by a group of children. The intention here is to specify a behaviour, that is based on a set of messages. The first two cases are called *coordination*. The intention of the behaviour specification is to describe how an EMS reacts to multiple messages from children for further internal coordination. For example, all children send their available resources. The coordination EMS decides on a fair utilization of the resources and responds with new set points. The specification of a coordination case can be quite complex, since the coordination behaviour might result in the application of extensive optimization algorithms. But it is sill important to define a proper coordination mechanism for an application, since it includes the specification of the proper input data and the proper output data. For instance, the presented market models or short-term planning model (see Section 3.1) can be seen as such coordination mechanisms. Any central control system, with only one particular decision making instance implements in fact this atomic behaviour. A hierarchical system has further interaction possibilities, as described below. However, we suggest to establish coordination behaviour on the lowest hierarchic levels. With that the necessity for *escalations* is reduced and the autonomy of more local EMS is increased within the hierarchy.

The next case of the lower set is called *aggregation*. The specification of an aggregation provides a mechanism to expose messages that combine the information from the lower layers. A simple example is the aggregation of the system state that describes the current power injection of an EMS. The input messages are in that case children specific power values. The aggregation mechanism is a sum (considering the losses if possible). The output message is the specific aggregated power value of the responsible EMS.

The remaining atomic case, the *state update* that is based on multiple children is similar as explained before. It receives several messages from the children and adapts its internal state based on those messages.

Trigger: Internal States

The final set for distinction of cases and the third triggering mechanism is determined by internal states. We explained in Section 4.2 that an EMS can be quite complex and have a significant internal structure as well. Internal events might therefore occur frequently. This case helps to specify how internal events affects the hierarchical applications. The four possibilities are depicted in Figure 4.12. The *state update* case has no external interaction. It is, therefore, not in the scope of the EMS specification. We briefly explain only the remaining three in the following.

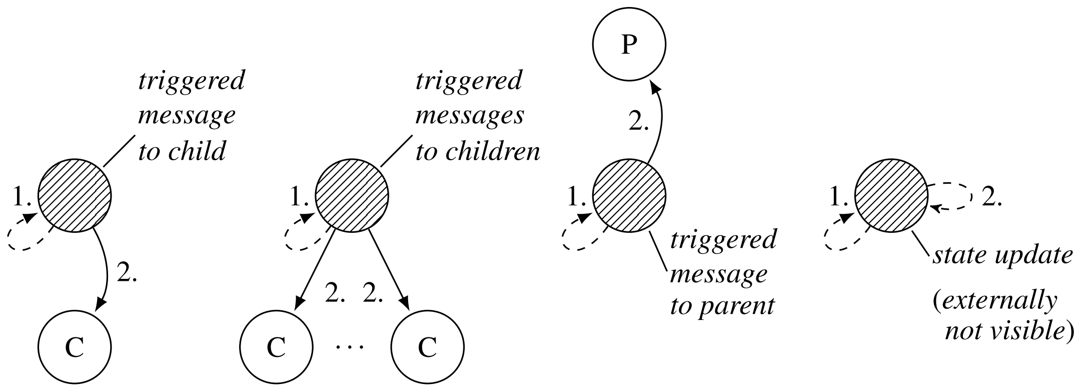

triggered message to child
1.
2.
C
triggered messages to children
1.
2.
2.
C
C
P
2.
1.
triggered message to parent
1.
2.
state update
(externally not visible)

Figure 4.12: The specification of the EMS behaviour based on internal trigger.

The *triggered message to child* and *triggered messages to children* cases appear when an EMS observes an internal state change that triggers an interaction with the lower hierarchical level. Such an internal trigger can be based on time, a sensor value or user's input. The trigger is sometimes used in the specification to define when an interaction starts in the hierarchy and defines which message is sent to start this interaction process. Since both cases are directed downwards the hierarchy, they can only be implemented by composites.

Finally, the *triggered message to parent* case describes all interactions that are started by an EMS towards the higher level. This case defines when the EMS begins an interaction with its parent. It specifies the internal trigger and the message that is sent upwards. The final case can be implemented by both, leafs and composites.

Distinction of Cases: Overview

The distinction of cases presents 16 distinct interaction cases in a hierarchic EMS architecture. Each case has its own individual input-output relation. Not all cases affect all types of components, composites and leafs. Not all of them are necessary to create an application to coordinate different EMS. For instance, a conventional centralized coordination is covered in our approach by the specific case, where children send data to the parent and receive a reply. Hence, this case in combination with one of the trigger cases to start the interaction is enough to reason about the conventional central coordination architecture. All other cases are not necessary. But, if EMSs are part in a hierarchy the interactions become more complex. Thus, we need to be aware that the hierarchic architecture as discussed by Moslehi and Kumar (2010); Benz et al. (2015); Howell et al. (2017) and others is much richer in their available interaction possibilities than presented with the flat-architecture approach. Also the system boundary of an EMS in terms of its input-output relation needs to be defined very clear, because each of the 16 cases requires particular input messages (that might include a lot of data) and particular output messages. However, there are several cases as the direct answer cases, that might be required several times for a particular application. Hence, even if there is only one specific direct answer case defined as an atomic case different instances of that particular case might exist. It is therefore not that simple. However, the distinction of cases is a helpful approach to develop specific hierarchic applications, as we demonstrate further below.

For coordination applications it is reasonable to combine several distinct interaction cases to establish a more complex interaction between the components. The exact combination depends on the application requirements. Further, some behaviours require that other behaviour specifications exist. Hence, the cases are implicitly related. Table 4.3 gives an overview of the described cases, if they are related and if they impact leafs or composite components.

The overview contains three major observations. Firstly, leafs require only a subset of distinct behaviour cases specifications, as they do not need to manage children. Composites on the other side, should have more defined behaviours, in particularly so that they can be used for coordination activities.

Secondly, the specification of some cases is marked as optional, because of two reasons. An internal state update is not visible for an external systems, since no information is exchanged. The specification of this case, therefore, helps to understand the internal behaviour, but it is not required to define an interaction with external component and define the structure of the message and its content. Therefore it is not required for any integration specifications and marked as optional. Further, we mentioned that a *triggered message to parents* is optional for composites. The reason is that a composite is usually only a supervisor of children. It does not have own devices that trigger alarms or violate some value limitation. Most interactions are based on children's input. Only some dedicated applications might require this specification, for instance reporting or a heartbeat. But these two cases can also be easily implemented based on a request from the parent, and not from an internal trigger.

Thirdly, the generic behaviour cases are interrelated and define a chain of interactions in a hierarchic systems. For instance, a top-down application could look like this: (i) At the top, an initial request is send to all children. As a consequence, another behaviour specification is required that describes the input based on a trigger from the parent. (ii) The intermediate levels could choose delegation as a desired behaviour. Delegation induces another parent triggered case specification. (iii) At the children level the application might require to update a state. This specification of this case ends the chain. Other applications would have similar interrelations and chains. Note, start points are always internal triggers. For instance, all kind of pull-based interactions can be implemented with a trigger that is based on times or on internal values. Push-based interactions can be started similarly. End points of chains are always state updates, without further interaction. This allows to analyse the specification of behaviours including a defined termination of an interaction sequence.

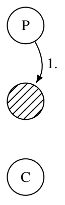
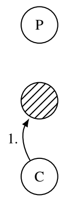
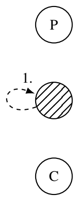
Trigger	Case	Leaf	Composite	Specification of follow up case
	P1: delegation	no	**yes**	→ 🟦
	P2: direct answer (to parent)	**yes**	**yes**	→ ⬜
	P3: state update (due to parent)	**yes**	**yes**	
	C1: direct answer (to child)	no	**yes**	→ 🟦
	C2: support	no	**yes**	→ 🟦
	C3: escalation	no	**yes**	→ ⬜
	C4: coordination	no	**yes**	→ 🟦
	C5: aggregation	no	**yes**	→ ⬜
	C6: state update (due to child / children)	no	**yes**	
	L1: triggered message to child / children	no	**yes**	→ 🟦
	L2: triggered message to parent	**yes**	**(opt.)**	→ ⬜
	L3: state update (locally)	**(opt.)**	**(opt.)**	

Table 4.3: An overview of the cases distinction for the behaviour specification of an EMS in a hierarchic architecture.

Summary

The core specification element in our generic hierarchic architecture is an EMS. It is part of a hierarchical structure and contributes to the overall system behaviour either as a leaf or a composite. Leafs deliver resources that are aggregated and managed by composites. Composites forward the aggregates of the resources upwards in the hierarchy, delegate requests downwards in the hierarchy and introduce an additional management level to support lower level components. The EMS behaviour in a hierarchic structure can be described with specific behaviour cases that need to be defined based on application requirements. The distinction of cases allows to reduce the complexity and describes a clearly specified input-output relation to define the behaviour of components.

The presented cases in a hierarchic EMS architecture represent a comprehensive collection of the possible EMS interaction patterns. The distinct interactions are not complex. They are simple input / output relations for the intended coordination activities. The simplicity is intended, since the SES vision of a holistic system with the large variety of different coordination possibilities is so immense, that we need a simplification in terms of the intended interactions to the very basic level, as presented here. Without the understanding of which interactions are beneath a hierarchic structure, the orientation in the development process is not easy and the system developers often limit themselves to central control systems, where the simple request-reply patterns dominate. Note, even if the atomic cases look rather simple, the combination of them in a hierarchic system allows a rich set of specifications that can be used for different kinds of coordination mechanisms for hierarchic EMS. We discuss more of those concrete and extensive examples in the upcoming chapters where we explain concrete implementations for different control scenarios. These implementations are different architecture instances of the presented generic architecture and cover several different behaviour steps that we introduce here. Those upcoming chapters cover a major work of this thesis, since they introduce concrete examples for SES architectures. But before we can start to describe those hierarchic architectures we need to provide the right technical environment to create SES applications that adhere to our approach with the generic hierarchic architecture.

4.3 Technical Support to Establish Hierarchical SES Architectures

The previous section introduces the generic hierarchic architecture on a conceptual level. To evaluate this concept for its technical feasibility and the expected benefits that are reached by this architecture, we need a sound technological framework, which is able to implement various SES applications that are then applied to different scenarios. This section presents this technological framework as the underlying technological infrastructure that has been created in the context of this thesis.

To study this impact of different SES systems and particularly to be able to create flexible SES applications we developed a co-simulation framework called Smart Energy System SIMulation (SESSIM). It combines three important areas for smart energy research: design, modelling and simulation of interactive EMS components (an architecture perspective), the influence to the physical system (power system perspective) and several interfaces to integrate real systems as (hardware in the loop and co-simulation integration) that test different scenarios.

The framework was initially presented in (Bytschkow et al., 2015). It integrates different technologies such as GridLab-D (a modelling tool for power flow dynamics), Akka (an actor based framework for large scale distributed applications), EclipseScada (a SCADA system for integration of remote components) and CIM (common information model for utilities). The co-simulation framework represents the fundamental vehicle for our analysis that we present further below.

4.3.1 Preliminary Considerations

Existing energy systems experience a paradigmatic shift towards more bottom-up intelligence in a system, where smarter and more flexible components provide additional services to the overall power supply system. Many research activities and technology providers delivered a wide range of new custom SES solutions during the last years, e.g. Fang et al. (2012); Koß et al. (2012); Becker et al. (2015); Mauser (2017). To measure their impact on the power supply infrastructure, energy models are usually used. However, it is often required that those models are enriched with real data, or even that the validity of those models is evaluated in real field tests. To do so, the models and coordination systems that use those models for creating control signals need to be combined with industrial systems. This is costly and time consuming.

As a preliminary and widely accepted evaluation step simulations are used for system analysis. Recently, simulations are becoming more combined, not only because the different granularity of the simulations, but also with real systems to validate the models, to reason about system requirements such as required time resolutions, the representation and interface specification for direct control, market driven approaches, demand response and flexibility capabilities as well as the integration of certain protocols. Instead of developing completely new solutions it is reasonable to combine the available simulations, tools and techniques. This was initially introduced by Hopkinson et. al. (2006) and Lin et. al (2011) in the context of SES.

For the interconnection of a simulation tool with real systems several possibilities exist. Available industrial protocols promote the usage of SCADA servers and clients. More recent applications also allow to integrate classical REST APIs, or systems with an OPC UA interface. When we consider the coupling of two different simulations we can use the functional mock-up interface, introduced by Blochwitz et al. (2012). Schütte (2013) proposes to use different abstraction layers for the available simulation tools, even though it always remains a challenge to understand and combine the systems, due to the different semantics, degrees of details, and usage purpose of the tools. Nevertheless, the approach of a federated framework helps with the goal of a co-simulation, particularly, if systems are analysed with respect to widely used energy model metrics that are well known and understood from the systems planning perspective. This also increases the acceptance of a co-simulation solution and helps system designers not only to simulate, but also test the solutions in a practicable environment and derive further insights about a possible impact.

4.3.2 Framework Requirements

The main purpose to develop a co-simulation framework is to create and analyse different SES solutions, where the hierarchic system is one of them. Instead of developing an independent, stand-alone solution, the goal is to use well established technologies that are well tested and effectively applied in the industry. Our co-simulation framework needs therefore to consider a set of requirements to enable such a design and analysis. This includes some essential modules for the framework that allow to model the physical world, the communication of between SES applications including the desired data exchange, as well as establish a connection to real systems. We explain the main requirements of our modules below.

SES solutions often involve complex algorithms for distributed control and decision making that is based on communication and information exchange. The communication may appear not only at the central location of the control center, but also within the distribution network and at EMS level within local areas or even buildings. This exchange of data is expected to influence the design of future power systems, since interactive components will become more and more an essential part of the infrastructure. Therefore, the *capability to model behaviour that relies on communication* - the EMS system behaviour, is the major requirement of the co-simulation framework. The behaviour models correspond to interactive EMS components that communicate between each other. For tests and simulation those behaviour models should be easy to refine. It also means that we should be able to separate the communication mechanisms from the behaviour of a component, i.e. the communication protocols should be interchangeable.

Another important requirement for the framework is the capability to represent the underlying physical system. The physical part emphasizes the production, delivery and consumption of electrical energy. Modelling and analysis activities for power systems have already a long tradition. Therefore, many commercial and research tools with comprehensive features are available, such as PSS/E, GRIDLAB-D, DIgSILENT PowerFactory, MATLAB's SimPowerSystems, OpenDSS amongst others. One requirement is therefore the *integration of* (possibly different) *power flow modelling and simulation environments*.

Utilities and system operators use different tools and create models that are adjusted to solve certain tasks and answer different questions. Depending on the purpose of the application, the level of detail varies and different models of the same system or time frames for the components are essential (Andersson, 2004). To overcome the burden of different models, a *standard representation with a good level of abstraction that supports export and import functionality to and from different tools* is necessary. If import or export of the standard model is not supported by the power flow modelling tool, the chosen representation of the model should allow to generate specific input models with a model transformation.

Additionally, multiple mapping conditions are required. Every physical node (e.g. production unit) should be able to map to a corresponding behaviour model. At the same time, not every behaviour model necessarily needs a physical component in the power network (e.g. SCADA or gateways for the electricity market). Sometimes the physical model is tightly coupled to a geographic location, e.g. using geographic information system (GIS) information. The geographical location influences the behaviour of some power components like photovoltaic or wind, and also provides a better indication to model the behaviour of power consumers. To use this information in the simulation framework, a *mapping should be possible from the environment model to the corresponding physical components and related behaviour models*, since alternatives for different mappings impacts the system behaviour. This is demonstrated by the research activities for deployment optimization of power components, e.g. by Akorede et al. (2010), and software components, e.g. by Aleti et al. (2013).

The next requirement considers the coupling between the behaviour models (interactive EMS components) with external components that add additional environment information and behaviour of real systems. Important environment information, such as weather data or market prices, affects the behaviour of the components and should be accessible in a simple manner. Therefore, a data base connection with a time series of historical date, or a mechanism to receive such information during runtime from an external source should be part of the co-simulation. Such an interconnection of the co-simulation is also important for the coupling of external components. Any modelled component in the simulation is an abstraction of a real component. To reduce the gap between modelling and real running systems, we require that *any behaviour model can be connected to a real system*. The concrete connection mechanism is not the main focus of the simulation. The preferred way is to use well established connections, such as clients for SCADA, REST or OPC UA. But also more experimental connections are fine, as long as we can connect to them over a specified network connection.

Beside the interconnection requirements, the real world environment and our co-simulation framework needs a clear understanding of time, to synchronize real-world and simulation behaviour. A mechanism to *separately define the application logic and the speed of the simulation execution* is therefore another important requirement. The splitting will also allow to slow down the simulation. This is also helpful for the visualisation of the simulation state. The interconnection with real systems imposes also a constrain on the maximum possible frequency of the time resolution and limits the analysis to higher level logic that includes tasks like planning, decision making and based on that setting the set points of controllers. Fast control loops that run on PLCs are not within the scope of the co-simulation framework.

In summary, to investigate SES systems in terms of their architecture, their communication, their power production and consumption and the impact of interactive EMS on networks, our technological framework requires to cover:

- Modelling of the behaviour of interactive EMS components
- Modelling of the physical power network, to calculate the power flow
- A mapping between the interactive EMS and the physical topology
- A standard model representation to support import and export functionality
- An interconnection with external systems
- A defined notion of time

4.3.3 SESSIM Framework

To carry out simulations with our architecture we developed the Smart Energy System Simulation (SESSIM) framework. It combines different tools that are executable at local machines or external servers. Instead of inventing new simulators, we use many components that have been developed over decades and are effectively used in the industry. A strict separation of the modules within the architecture allows to reduce the complexity and use (proprietary) components that are developed and updated independently. The architecture of the framework is depicted in Figure 4.13.

The system model modules is the main component for the inputs of the co-simulation and supports different representations. It consists of different sub-modules including an importer for power system models, an importer for EMS topologies and behaviour models, environment information, and a mapping module. When starting a simulation, the systems model importer is

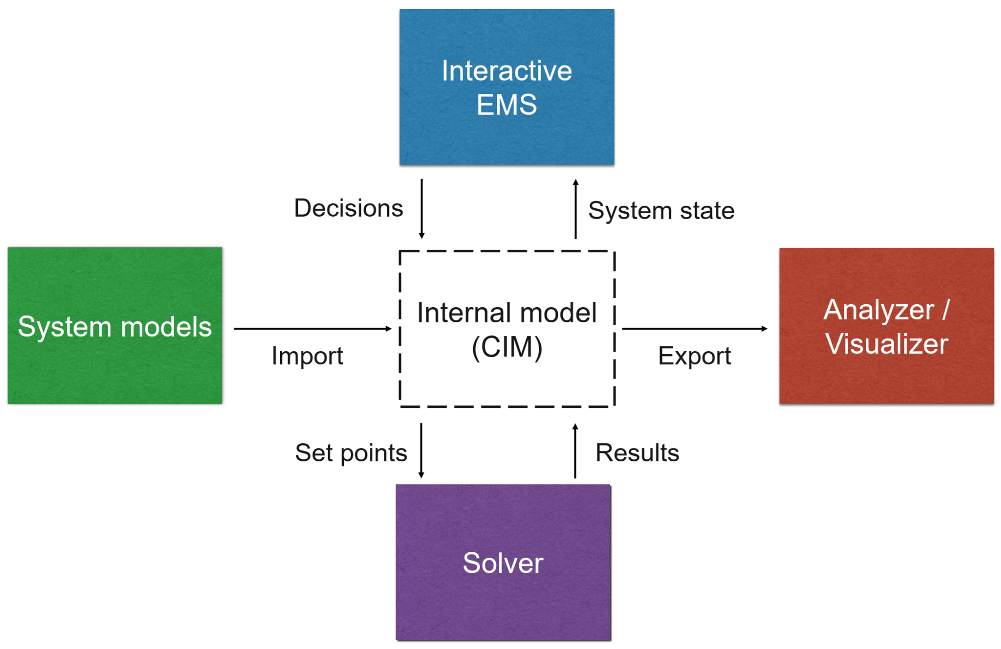


Figure 4.13: Architecture of the SESSIM framework

started to populate the simulation and load all required data into the execution environment. The power system importer creates a CIM-based model that is also used as the internal representation of the physical system during the co-simulation. CIM consists of a number of European and international standards. IEC standard 61970-301 contains the components of a power system at an electrical level and the relationships between them. IEC 61968-11 covers the areas of asset tracking, work scheduling and customer billing. Relevant for exchanging data between participants of electricity markets is IEC 62325-301. We use IEC 61970-301 and IEC 61970-456 for our co-simulation. IEC 61970-456 extends CIM with capabilities to store solved power system data (power flows). For data storage CIM RDF is used as described in IEC 61970-501.

The system behaviour in terms of the EMS input - output is modelled as a set of communicating components that mostly represent EMS inside the interactive EMS module. In our model, we assume that every component represents an independent entity. A component receives and sends messages. It designate its behaviour based on the received message by changing its internal state and sending other messages. With that every component acts independently without globally shared variables. This allows to model inherent concurrency of each EMS. Furthermore, it enables to implement the application logic independently of the physical model and instantiate it at run-time with parameters based on the mapping of EMS and physical components. A connection to external components is achieved by giving each EMS representative one or more clients including REST and SCADA components. Both, the REST and SCADA clients connect to external servers and transmits the received information to the other simulated EMS components. It also allows to send messages from the simulated system to the external server systems. To keep the system synchronized our interactive EMS module periodically checks the timestamps of the external connections and adapts the internal simulation clock. The implementation details are presented below.

The power system simulation module is performed using a solver that calculates the system state based on the decisions taken by the EMS components and the measurements transmitted by the SCADA system. The decisions and measurements represent set-points for the power simulation. The power system solver returns as a response the resultant values of physical properties of the power system, such as voltages and power flows. The values are stored in the CIM model and read by the EMS components. The mapping between the different modules is established using unique identifiers of interactive EMS components, i.e. the *mRID* identifier according the CIM standard.

Finally, the time series of the results is available at the analyzer / visualizer module that stores the results a file or a data base, depending on the choice of the system engineer. This allows to process this data by any desired visualization framework or to perform further evaluations.

Interactive EMS Module (System's Behaviour)

The behaviour of software intensive systems for SES can be modelled in different ways. Depending on the questions of interest, the model of computation can be, among others, event-driven (Godfrey et al., 2010), agent-based (Hopkinson et. al., 2006; Lin et. al, 2011), actor-oriented (Lee et al., 2003) or stream based (Hackenberg et al., 2012). For the implementation of the interactive EMS module we have chosen the akka toolkit[9]. It provides a combination of the actor approach with streams. akka is an open source toolkit that is widely-used in the industry for highly resilient and reactive systems. It is very scalable and allows a distributed implementation of components that run on different servers. akka runs on the Java Virtual Machine (JVM) and supports applications written in JAVA or SCALA. We define SES components with a behaviour model for actors that are able to communicate and mapped to power system equipment. In the following we shortly explain the most important implementation constrains, to demonstrate how co-simulations are created and which basic functionality they provide. The initial implementation work of the module was carried out together with Mack (2014) and extended later on.

Actors in Sessim

Our co-simulation framework is based on actors (see Chapter 2). They create a hierarchic topology that is the basis of our simulation framework. Actually, every simulated SES component is an actor with an own behaviour. We call those actors `SessimActors`. In addition we have two actors for timing effects and as a coupling mechanism to other modules, in particular, the power flow simulation module. They are called `SessimSupervisor` and `SessimMonitor`. We illustrate the relationship of all actors in Figure 4.14.

The `SessimSupervisor` is responsible to trigger the execution of the simulation in Sessim. It initializes the first layer of the simulation actors according an topology model. The topology model is prepared as part of the simulation definition. With the initialization, the supervisor becomes the parent of the first layer actors. The first layer actors spawns then the next layer of actors as its children according the topology model as well, and so on. After spawning, the original actor in akka is always as a supervisor that handles the failures of its subordinates. We use this natural akka convention for our simulation to create a generic hierarchic system model. Beside initialization, the `SessimSupervisor` controls the time step execution.

[9] `http://akka.io/`, developed by Lightbend Inc., last accessed in March 2020

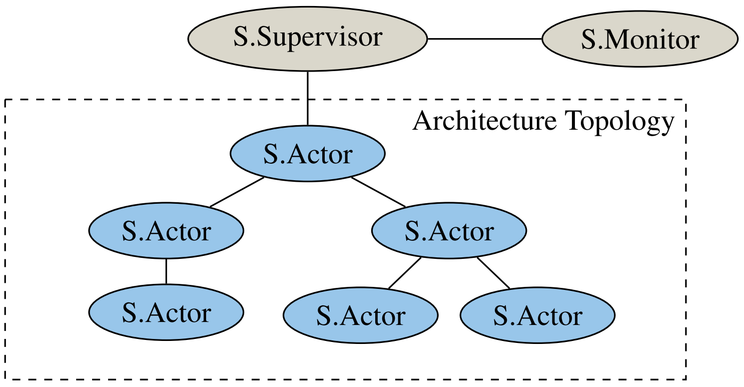


Figure 4.14: Actors in Sessim and their internal hierarchy.

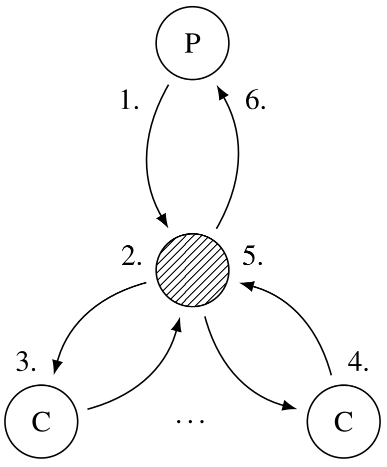
The `SessimMonitor` interacts only with the `SessimSupervisor`. It is the only component that has an externally available message inbox to establish a connection to the actor system from the outside. It provides an API to control the simulation from external applications. Additionally, it monitors the time of the simulation and is used to adjust its execution speed. The `SessimMonitor` is also used to start external tools or systems, whenever the simulation needs those. In our case we used it to trigger GridLab-D and our own power flow solver, that we will introduce later.

The `SessimActor` components are responsible for the main simulation logic. Every `SessimActor` has its own behaviour. Therefore, `SessimActors` are representing EMS that have an application specific logic and a dedicated interface to exchange data. The individual behaviours are adapted flexibly to cover different SES scenarios.

SessimActors' interactions

The behaviour of a `SessimActor` follows the interaction pattern that is based on the atomic steps that we presented in the previous section. This interaction pattern is common for all `SessimActors`. This facilitates the creation of simulations and helps with the development of scenarios in a quick and clean way. We show the basic interaction pattern in Figure 4.15.

The interaction pattern consists of a number of steps that are implemented in Java as abstract methods of the general `SessimActor` behaviour. Every `SessimActor` awaits a request message in its first step. The request message can contain any information. This flexibility is used to request services or send set-points or prices. It is therefore well suited to create message based interfaces. If no content is desired, the request is just the forwarded time step from the SessimSupervisor. After receiving a request message, the `SessimActor` prepares its own request, and sends it to its children. After sending the request, the actor waits until all children have answered. Then, the `SessimActor` processes the answers, which again have an application specific content. Based on the answers it makes a decision. It aggregates the answers and executes additional actions, e.g. stores the answers in a data base or contacts a remote application. After the decision making the `SessimActor` notifies its parent with an answer. This hierarchic combination of the delegation and aggregation behaviour allows to model all the behaviour cases according the generic hierarchic architecture from the previous section.



1. receive request(s)
2. prepare requests
3. send requests to children
4. receive answers
5. make a decision / prepare answer(s)
6. send answer(s) to parents

Figure 4.15: The basic interaction pattern of SessimActors.

Connection to external components.

The Sessim framework offers multiple possibilities to establish real-world connections, as akka is designed to build reactive applications in a web based environment. Therefore, we have a rich set of libraries to interact with external components over HTTP(S), REST, sockets or similar interfaces. Our actors can start their own clients or servers. This allows us to connect any actor to a remote application. If we want to interface with applications from the utility domain, we use additional technologies in Sessim. As an example for the IEC 60870-5-104 we demonstrated how to integrate Eclipse NeoSCADA[10]. It is an an open source SCADA implementation with a client-server architecture that is used in industrial applications. We used different ways to demonstrate the integration. The first option with the akka framework is using the *inbox* of akka. A second way is to create a client or server instance directly from an actor and connect it to an external component. In this case the actor sends a message directly over it own client or uses its server to update the data. We show both possibilities in Figure 4.16. We preferred to use the second option, where each actor has its own interface after the project, since it offers more flexibility as the first one.

Intermediate Model (CIM)

The intermediate model is the representation of the physical topology. It allows the coupling of our interactive EMS module that handles the message exchange between EMS with a solver for the electrical power flow equations. The coupling is achieved with a mapping of every actor to an object of the internal model.

The internal representation of the system is implemented according the Common Information Model (CIM). CIM is based on standardisation activities for a better interoperability of utility systems. It particularly addresses the exchange of topological data of power networks between different system operators, that rely on SCADA systems from different vendors. Initially, CIM

[10]`https://www.eclipse.org/eclipsescada/`, last accessed in March 2020

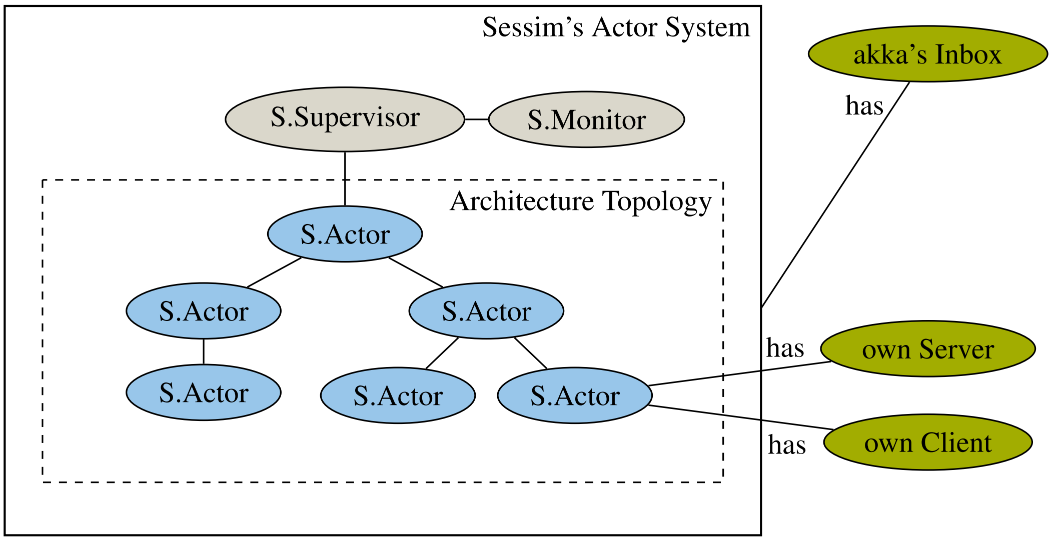


Figure 4.16: Different possible connections with external components in Sessim.

was a development of EPRI[11] to standardize EMS interfaces Uslar et al. (2012). But it has grown beyond the original scope and has become an international standard. CIM contains the standards IEC 61970-301, IEC 61968-11, IEC 62325-301. It has also received a translation into national derivatives (DIN EN 61970-301, etc.). For the historical development of CIM and we refer to (CIM, 2015; Uslar et al., 2012).

CIM follows an object oriented approach to structure its data. It provides a specification to model energy related equipment and allows to encode a different granularity of the system. Such different levels of details are illustrated in Figure 4.17. Concerning the level of detail CIM is a so called node-breaker model. A node-breaker model does not only describe the conducting elements of the power system but also models additional nodes e.g. open switches. In contrast to that a bus-branch model as it is typically used in power flow simulation tools only describes the conduction elements of the network. As a consequence it is necessary to map the elements of the CIM model (which we would typically get out of a network planning tool) to their bus branch representatives. This is achieved through aggregation, omitting and direct mapping of the CIM elements and is described in further detail in (McMorran et al., 2004). Once the mapping is done the bus-branch representation is processed for the desired power flow solver.

Instead developing an own internal data model, we believe that standard confirm models provide several advantages. They help with the interoperability and allow to reuse available software components and libraries of third parties. For the implementation of CIM we used an *ecore model*[12] of the CIMtool[13]. The *ecore model* is used to generate the required objects such that they are consistent to CIM standards. Using an *ecore model* that is developed by international experts reduces the risk of inconsistency. Further, we are able to reuse existing models and directly use import and export functions of commercial tools or connect our co-simulation framework. If

[11]Electric Power Research Institute

[12]Ecore models are used as (meta-)models in the Eclipse Modelling Framework (EMF)

[13]CIMtool is an open source tool developed to support the CIM standard. `http://www.cimtool.org`

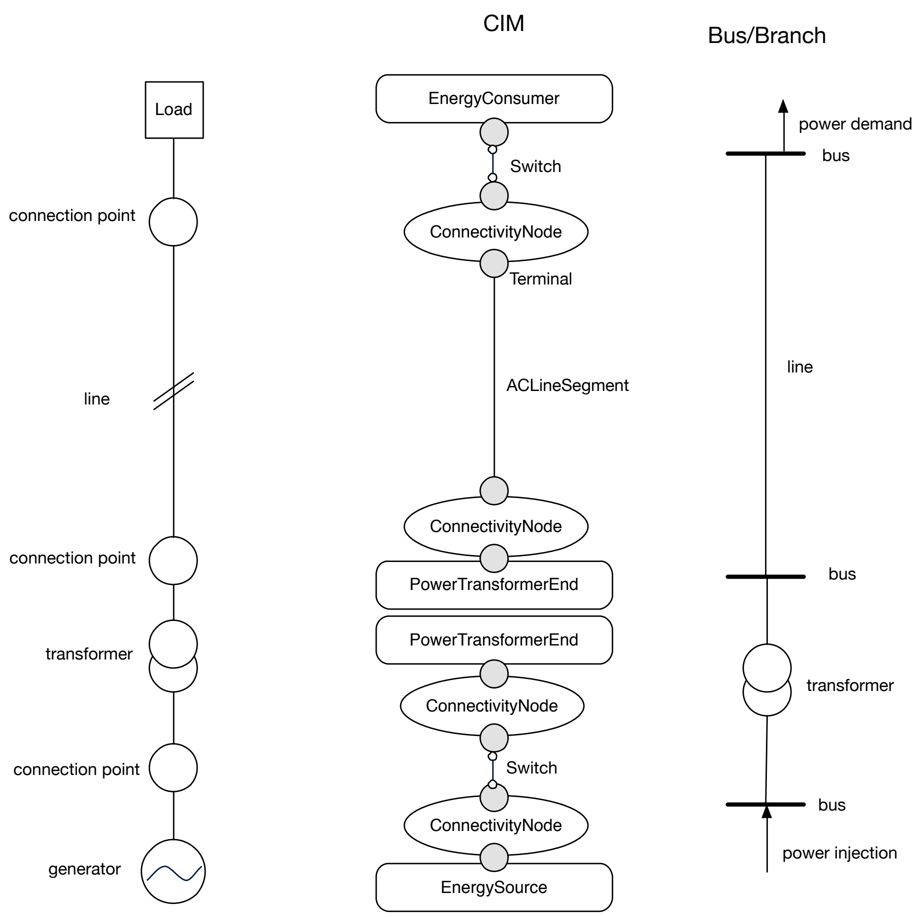


Figure 4.17: Details in a general model, CIM and a bus/branch model (Zellner, 2014).

we want connect tools that do not support CIM, we generate the required models, for which CIM provides a good level of abstraction.

Physical Representation and Solvers

Depending on which power flow solver is used, the bus-branch representation of the model must be represented in a form that is readable by the solver. We have implemented two model transformations. One for the OpenSource tool *GridLab-D*, a power system modelling and simulation tool, and another one to a *matrix/vector format* that is used to solve the power flow model internally with a Newton-Raphson method.

GridLab-D

For GridLab-D the standard input format is a text file according the glm format. For example a simple bus (node) is defined in the glm format as shown in listing 4.2 and a branch (line) as defined in listing 4.1. For more examples we refer to the GridLab-D documentation[14].

[14]`https://www.gridlabd.org/`, last accessed in March 2020

Listing 4.1: glm line notation (branch)

```
object underground_line:1 {
    name lineA;          phases AN;
    from node:1;         to node:2;
    length = 123.4;
    configuration line_configuration:1;
}
```

Listing 4.2: glm node notation (bus)

```
object node:1 {
    name node1;
    phases AN;
    nominal_voltage 7200;
    voltage_A 7200+0.0j;
}
```

As a consequence, it is necessary to render the bus-branch java objects to a text file that is compliant to the glm format. Since we already reduced the model complexity to a bus-branch level, we only use the node (buses), line (branches) and transformer objects of GridLab-D for the topology. For the power injections we use the load objects that are connected to a node over a line. The abilities of GridLab-D to simulate changes in the network topology (e.g. closing and opening switches) are not required. This part is already handled by the CIM model. After the transformation of the model to a glm file the calculations are executed using GridLab-D. The results are returned by GridLab-D as xml which can be read and parsed. Finally the results are mapped back to the CIM objects they originated from and stored in CIM using the IEC 61970-456 CIM extension.

Matrix/Vector format

Interfacing to GridLab-D from JAVA is rather complex and slow due to the file I/O in each simulation time step. Therefore an implementation of a own solver was undertaken together with Zellner (2014). The solver is in many aspects very similar to common solving frameworks such as MatPower but works directly on the JAVA representation of the CIM. Input for solving the power flow problem using the Newton-Raphson method (see also Section 3.1.2) are the admittance matrix $\mathbf{Y}$ of the network as well as the bus voltages $\mathbf{V} = [V_1, \dots, V_n]^*$ for the n buses and the power vector $\mathbf{S} = [S_1, \dots, S_n]^*$ that represents the desired power injections and demands at the n buses. Note, all elements of the vectors and matrices are complex numbers. The values are related as shown in (4.1) - (4.3), where $\mathbf{I}$ denotes the electric current and diag being the operator constructing a matrix from a vector with the elements of the vector on its diagonal.

$$\mathbf{S} = \text{diag}(\mathbf{V}) \cdot \mathbf{I}^* = \text{diag}(\mathbf{I})^* \cdot \mathbf{V} \tag{4.1}$$

$$\mathbf{I} = \mathbf{Y} \cdot \mathbf{V} \tag{4.2}$$

$$\mathbf{S} = \text{diag}(\mathbf{Y} \cdot \mathbf{V})^* \cdot \mathbf{V} \tag{4.3}$$

The physical relation (4.3) establishes a set of $2(n-1)$ non-linear equations, since for each bus, except the slack/reference bus, the complex equation has a real and an imaginary part. Note, that in those equations, some values are given, others have to be calculated. The solution gives the current physical system state. The approach for solving the equations is an Newton-Rapshon iteration procedure. With (4.4) we describe a deviation of one calculation from the desired solution. The iteration is defined in (4.5).

$$\mathbf{x}_{mis}(\mathbf{V}) = \text{diag}(\mathbf{Y} \cdot \mathbf{V})^* \cdot \mathbf{V} - \mathbf{S} \tag{4.4}$$

$$\mathbf{V}_{k+1} = -\mathbf{J}^{-1}_{\mathbf{x}_{mis}}(\mathbf{V}_k) \cdot \mathbf{x}_{mis}(\mathbf{V}_k) + \mathbf{V}_k \tag{4.5}$$

Here, $\mathbf{J}^{-1}_{\mathbf{x}_{mis}}(\mathbf{V}_k)$ is the inverse Jacobian matrix evaluated at $\mathbf{V}_k$ for our power flow function (4.4). If the deviation is less then a desired threshold, we calculate the resulting bus voltages $\mathbf{V}_{res}$. Using $\mathbf{V}_{res}$ we calculate the power injections $\mathbf{S}_{res}$ within the system.

$$\mathbf{S}_{res} = \mathrm{diag}(\mathbf{Y} \cdot \mathbf{V}_{res})^* \cdot (\mathbf{V}_{res}) \tag{4.6}$$

With the input variables for the solver being directly generated from the CIM objects it is possible to map the solver results back onto the CIM objects and store them CIM conform.

Verification of the power flow solver.

The implementation of an our solver was verified to demonstrate the correctness of the results. The classical approach to show this is to apply the solver a well known problem set, for which tested results are available. The power flow problem has several of those examples. They are recognized and verified by IEEE. The test cases with the verification is provided in our extended examples code of our open source implementation[15].

Synchronisation of Modules and Coupling with External Systems

We described the three major modules of our co-simulation framework. The modules are independent of each other and need to be coupled for a proper interaction. To couple the modules different approaches were investigated. One possibility is to run simulations independently and in parallel. Their coupling is then based on time frames. Whenever, both simulations are complete, the data is synchronised. This coupling was used by the EPOCHS framework (Hopkinson et. al., 2006). In this approach both simulators synchronizes with a mediator after a pre-defined simulated time frame. It has been shown that this coupling introduces time accumulating errors (Lin et. al, 2011), in particular when events occur between the synchronization points, that should affect both simulators. Therefore, a globally synchronisation in a discrete, event-driven manner is preferred (Lin et. al, 2011).

We have chosen a discrete, even-driven synchronisation for our framework as suggested by Lin et. al (2011). For the interaction with external components several conditions are given. First, we need to be able to handle asynchronous communication schemes, over different communication channels. The communication latency is not known a priory, but a typical TCP/IP communication is in a range between several of milliseconds up to some seconds. Since we need to calculate the actual system state of the interactive EMS and the power system, which might contain hundreds or even thousands components the synchronisation cannot be expected to have milliseconds range. Given these conditions our a co-simulation framework in not intended for fast control schemes and real time systems. But we can establish co-simulation of coordination scenarios with the models presented in Chapter 3.

To allow asynchronous communication with external systems coupling is established as illustrated in Figure 4.18. First, our interactive EMS module supports asynchronous communication with external systems. For that each actors starts dedicated server instances that receive messages and respond to messages in parallel to the simulation. We continuously buffer all the incoming

[15] `https://github.com/SES-fortiss/SmartGridCoSimulation`, last accessed in October, 2019.

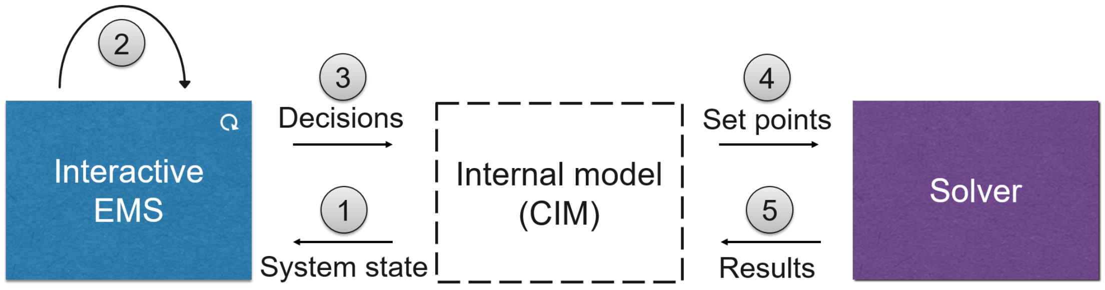


Figure 4.18: Coupling cycle within the co-simulation framework.

messages from external systems. This is denoted with the reload label. Beside that we couple the modules with a cycle that determines the next system state based on the previous one. The starting point is the actual state that is stored within the CIM model. The `SessimMonitor` starts the process with a broadcast of the time step. At step (1), every EMS actor reads the current state from its linked component in the CIM model and from buffers of the external components. This transfers the current system state knowledge to every actor. Then decision making takes place (2), as explained in the previous section. This can involve a complex message exchange. During the decision making process the actors can also request data from external components to access remote information. After each EMS has decided their new state (e.g. a new power injection value of a generator or consumer) the new set points are stored in the CIM model (3). This updates the state of the physical model. The power flow simulation is executed with new values (4). It determines the new system state of the timestamp. The results are stored in the CIM model (5). The interactive EMS module considers them during the next iteration.

Testing of External Systems with SESSIM

The framework is not only used to study EMS in combination with physical networks, but due to its excellent capability to create distributed independent systems it is a good reference to develop further energy related use cases and test external systems. We used Sessim to create small trading applications to integrate it with several distributed ledger technologies such as blockchains. The integration with Ethereum was implemented and described by Thut (2018). The integration with Hyperledger was implemented and described by Lumani (2018). Further tests were undertaken with IOTA and the blockchain platform from the Energy Web Foundation (EWF). All those studies contributed to a good comparison between the blockchain technologies and presented by Bajpai and Duchon (2019).

Further, the approach in Sessim with the definition of messages and interfaces leads to the idea to use that data also for Machine Learning approaches. To show that this is the possible Andres (2019) used our approach and trained different Reinforcement Learning algorithms. He compared the A2C/A3C (Asynchronous Advantage Actor-Critic) (Mnih et al., 2016), PPO (Proximal Policy Optimization) (Schulman et al., 2017) and DDPG (Deep Deterministic Policy Gradient) (Lillicrap et al., 2015). This shows the flexibility and maturity of the framework. It also underlines again that the strength of our approach is not only the investigation of power flows in combination with an actor based communication, but particularly the combination with different kind of technologies, such as blockchains, machine learning and integration of real systems.

4.4 Summary

To realize the hierarchic composition of EMS and establish coordination within such compositions, work on the architecture is as important as work on technologies or coordination models. In this chapter, we present the basic foundation for the generic hierarchic architecture from different perspectives. We provide several steps to derive such architectures.

We start from requirements that we observe in today's energy systems where we assume that EMS are individual components that should be interconnected. Further, we present a systematic approach how an architecture can be developed on top of those requirements using available best practises in the form of well-known architecture design patterns from the software engineering perspective. We present a careful selection of available architecture design patterns that can be combined to cover the list of the aforementioned requirements.

Furthermore, the architecture is not a pure collection of architecture design principles, but the behaviour of such a system is quite important for architecture specifications. To understand the effects of the hierarchic composition for component behaviour, an abstract model of for the behaviour of individual EMS is introduced. It demonstrates the possible interaction patterns of an EMS with the help of a case distinction. We show that 16 basic distinctions exist. The cases are interrelated, as some cases require the existence of other cases. Further, the cases are neither singletons, since multiple instances might exist, nor all cases are required for EMS leafs and EMS composites. However, the introduced approach is a step towards the design of architectures as envisioned by Moslehi and Kumar (2010), Grijalva and Tariq (2011), Benz et al. (2015) and Howell et al. (2017), because it helps to reason about the core components that are necessary for coordination, their interfaces and their behaviours.

Further, we present a technical framework that has been developed during this thesis to create concrete instances of the hierarchic architecture. The technical framework is the Sessim co-simulation and available as open source. Sessim uses a hierarchical structure with actors that represent interactive EMS components. All actor components run in parallel. Their hierarchic structure allows to decompose the system into individual behaviour models for each EMS and establish architectures with distinct components, distinct interfaces and distinct behaviours for EMS. We use them to create specific hierarchic architectures for SES. Those specific architectures focus on the interaction between multiple EMS components and represent the network perspective is represented by the co-simulation framework. In addition, the actors can establish connections with real systems. This allows to use the environment to study the interaction with EMS systems and also simulate their impact on the network level.

Finally, there two additional remarks to this chapter. First, the concept of a hierarchical architecture is not only applicable to the interconnection between EMS. It can also be applied for internal EMS sub-components as well. We show this in Appendix A. Second, the framework is very well suitable to create an environment for additional SES studies, for instance to interconnect with distributed ledger technologies, such as blockchains (Thut, 2018; Lumani, 2018) and also to create input data for Machine Learning systems (Andres, 2019). In the following two chapters we present two concrete instances of the hierarchic architecture for the coordination of EMS.

5 | Hierarchical Architectures for Virtual Power Plants

This chapter describes a case study that we performed with several industrial partners, among them a SCADA provider and a utility company. The goal of our collaboration was the development and the technological demonstration of a more flexible VPP that supports interactive clusters of components. With this case study we investigate and clarify the meaning of clustering and hierarchical structures for the coordination of energy systems. This includes the meaning of system boundaries and system interfaces. Further, we particularly address the first research objective stated in Chapter 1. We demonstrate the technical feasibility of our concept and validate the chosen architecture design patterns from Chapter 4. Note, that some product names and logos used in this case study are protected brands of the industrial partners.

5.1 Context of the Study

The liberalization of the energy markets combined with the increased number of distributed renewable energy sources lead to the development of the VPP concept during the 2000s as described by (Santjer et al., 2002; Willems, 2005). Renewable energy sources are often too small and volatile to meet the minimal trading requirements of the energy markets and too volatile to guarantee a fixed schedule for generation. However, they are an excellent option to provide power with less carbon emissions and increase the energy utilization within local networks. To enable the participation and comply with the requirements of the energy market, the decentralized components are bundled virtually. In this bundling, local renewable systems monitor their production and communicate the data over gateways to a centralized software system. The software system collects all those values, calculates forecasts and places bids at the market (see also Chapter 3). The software systems is, from the technological point of view, an EMS with the functionality to dispatch the connected units as well as monitor and control them at runtime (Ghavidel et al., 2016; Nosratabadi et al., 2017).

The concept has already turned into existing technological solutions. Multiple providers offer it as a service to owners of distributed resources such as solar, biogas, wind, water, geothermic and emergency power supply units, combined heat and power as well as controllable demand. Examples for such providers are: Statkraft (a norwegian power producer and Europeans largest VPP in 2019), Next-Kraftwerke (a large German VPP provider), Energy2market (also a German VPP provider) and SWM M/Partnerkraft (a VPP product of the German utility company Stadtwerke München GmbH).

Today, a VPP, or more precisely the software that collects and processes data to aggregate the components and interact with the market, is created on top of a system that follows the SCADA architecture with a flat, central hierarchy. By flat and central, we mean that there is a core system to integrate all other components. This is schematically shown in Figure 5.1. Of course, the system can be replicated to provide more reliability, but the information and the decisions for the operation are done by one central component.

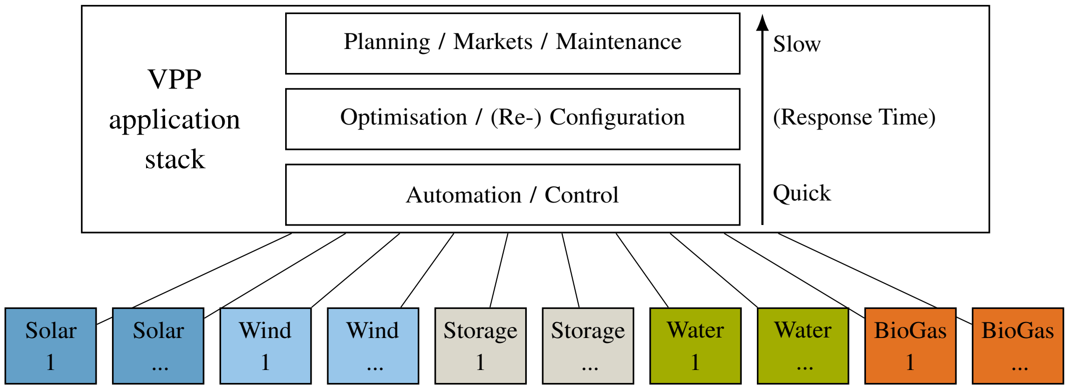


Figure 5.1: Typical architecture for a virtual power plant with common applications sorted by expected response times.

The central approach for VPPs has several advantages. The primary advantage is the availability of data in a central place. It allows different applications to use that data. It also eases the monitoring and control by an engineer at the control station. Additionally, the system provides a good reliability, due to the replication capabilities and backup possibilities. It is usually also heavily secured using different firewalls, virtual networks with dedicated access and a physical restriction. Today, a variety of energy system applications use the central approach. We refer to the literature to show the variety of energy related applications, for instance to Fan and Borlase (2009), who provide a good overview in their work.

The centralization, however, introduces also disadvantages, especially when the number of connected components becomes larger and more divers. Particularly, the integration of new components and interconnection of different EMS is challenging. Today, the integration process for new components requires a lot of manual work. VPP providers usually offer a preconfigured box to connect remote power generation components. Customers connect the box to the controller of the generation unit, which is usually some kind of a local Programmable Logic Controller (PLC). The controller establishes a connection to the box over a MODBUS, PROFIBUS, TCP/IP or similar interface with industrial protocols, where data and control signals are available. Depending on available interfaces, the integration can be more or less difficult. After that, the box communicates to a central back-end system of the VPP. After the technical interconnection of the component with the back-end system, the (central) applications are configured with the new components according their operational settings. This means that the parametrisation for control and optimization have to be adjusted.

Establishing the integration with such VPP boxes has side effects. Firstly, the installation uses the local device interfaces that have no abstractions of their data. In consequence, the control of the device is directly accessible with the box, but encapsulation principles to guarantee privacy or information hiding cannot be applied for the customer. Abstractions are however necessary, to describe the capabilities of the component and for data aggregation. Secondly, the box is a proprietary product that is linked to a dedicated VPP system. Whenever the customer decides to change the VPP, the box needs to be exchanged. This is quite costly, as the physical boxes and the system configurations needs to be re-adapted. In a flexible SES environment, where many small components interact, this change has to be automatable. So the box needs either to be independent of the provider and provide standardized interfaces and services, or the device manufacturer has to provide this interface. Such approaches are for instance driven by the German VHPready Consortium with their VHPready (Virtual Heat and Power Ready) standard (VHPready, 2017), the EEBUS Initiative e.V. with their EEBus SPINE (Smart Premises Interoperable Neutral-message Exchange) standard (EEBus, 2018) or the OpenADR Alliance with their OpenADR (Open Automated Demand Response) standard (OpenADR, 2013). Thirdly, in modern energy supply systems generation units can be locally combined to optimize the internal energy supply from the economical or the environmental perspective, for instance a photovoltaic system with a battery or a local heat pump. For the local optimization, different control strategies and corresponding system configurations exist. The direct control with a box interferes with the local configuration. This leads to a less stable and less configured local system. The described challenges are related to the system architecture. In a divers and large environment that is present with current distributed systems, the central approach requires additional flexibility to integrate not only single components, but also substructures, and abstract representation of devices.

Further, the VPP operation is developed to establish a schedule, which is influenced by the market. To achieve the scheduled production is not as easy as for a conventional power plant, due to the volatility of the renewable energy resources. Due to increasing number of distributed renewable energy resources, VPPs are rapidly growing. Current VPPs bundle hundreds or even thousands of components. However, many components are only connected to monitor their production. This helps to improve the prediction quality and the market bidding. Only a little number of components can also be directly controlled. This allows to provide a smoother production schedule. In addition, control capabilities are often limited due to a number of reasons. For instance, solar and wind generators have a prior rank from the regulative perspective, thus they feed in as much available electrical power as possible as long as stability is not threatened. Run-of-the-river water plants often avoid hydropeaking due to ecological reasons for fishes. Biogas and combined heat and power (co-generation) units prefer a rather constant operation, since their generator is optimized for a certain power output and preferably generates constant heat during the cold times, while electrical power is a secondary co-product. Therefore, an adaptive scheduling and optimization is often used for a minimal subset of the components. However, with the increasing number of storage components, electric vehicles or heat pumps more types of flexible components will be accessible in the near future. Also the collection of renewable energies strongly increases, which increases the necessity for their control. This allows more and more control and optimization being carried out. To reduce the optimization complexity a decomposition of the VPP into smaller optimization problems from smaller clusters might be helpful. Also the integration of single components to one particular cluster leads to a simplification, since only single clusters are changed, not the complete VPP. This reduces the risk of adapting the controllers of the complete system and increases the modularity.

5.2 Study Goal and Research Questions

In this section, we present a more formal version of the study goal and derive from that goal further research questions to guide our work in the following.

5.2.1 Study Goal

We stated in Chapter 1 three research objectives. This case study tackles the first one. It is to **determine the essential specification elements for the hierarchic composition of systems to achieve a hierarchic coordination of energy systems.** We paraphrase the research objective into a more formal study goal using the goal definition template as proposed in Wohlin et al. (2012):

We analyse	*energy coordination systems*
for the purpose of	*characterization, understanding and specification*
with respect to	*hierarchic composition*
from the viewpoint of	*software developer and system integrator*
in the context of	*virtual power plants.*

The question for composition of software systems is always related to the system interfaces and system behaviour. Hence, with this case study we particularly evaluate those two aspects. For the technical development of such systems we further aim to validate the feasibility of the architecture design patterns presented in Chapter 4.

5.2.2 Research Questions

Based on the study goal we derive the following research questions.

RQ1: How does the hierarchical concept affect the integration of interactive EMS and the architecture of composed energy coordination systems?

With this question we aim to investigate the hierarchic composition of EMS. We are particularly interested in the integration of EMS and related devices into distinct groups that are then integrated into larger VPP groups and so on. By answering this question we want to understand, if we can create a hierarchic composable VPP and what it means for the system interface and system behaviour. To have a clear understanding we make the composition application specific with a typical coordination functionality of a VPP. This allows us to demonstrate how the hierarchy affects the design of energy coordination systems in a constrained context.

RQ2: Does the hierarchy support different energy coordination mechanisms?

This research question targets the understanding about the functionalities of VPP clusters. We aim to analyse, whether our approach is limited to some particular coordination algorithms, or if it can be applied in a broader sense for different applications and services in a VPP.

RQ3: How do the proposed architecture design patterns help us with the development of such coordination systems?

By answering this question, we aim to validate the architectural design patterns that we described in Chapter 4. Specifically, we want to demonstrate the technical feasibility of a hierarchic VPP system application that uses different distributed components as we see in real systems and interconnect those systems using an architecture that utilizes the proposed architectural design patterns. The idea is to show, that the architecture design patterns help and simplify the creation such hierarchical systems.

5.3 Study Design

To answer the research questions we show with an experimental implementation a coordination example of a demonstrator that uses a hierarchic structure.

5.3.1 Selection of the Coordination Goal

One major goal of a VPP is to enable to participation of smaller renewable energy systems at the energy market to increase their revenue. Therefore, the VPP places bids at the market according to weather forecasts and availability constraints of its components. The bidding process takes place as described in Chapter 3. After the bidding process, the VPP receives acceptance notifications of its bids together with the expected market price. The notifications are available quite ahead of the operation time, for instance one day. They represent the expected operation schedule of the VPP. When the time approaches the actual operation, the forecasts are updated. This might lead to the identification of expected deviations from the schedule. To reduce those deviations the VPP operator is allowed to trade at the intraday market as well. This can be done up to approximately five minutes before the actual operation. Once, the limit of five minutes exceeds, the VPP is supposed to deliver its contracted schedule. However, the VPP might still experience certain deviations, either due to wrong forecasts, missing options at the intraday market, or when components are unexpectedly shut down. Hence, it flexibly needs to adapt its operation by providing the desired schedule.

In this case study we approach this necessity for control with a coordination using *flexibilities*. The term *flexibility* is quite often used in the power systems domain to describe the possibility to adapt the operation due to external changes as described by Huber (2017). Consequently several *flexibility models* exist, such as presented for instance by Huber (2017); Zade et al. (2018); Nalini et al. (2019). Currently, even markets for flexibility are investigated that follow a similar approach as Nalini et al. (2019). In our case study we use the term flexibility to describe that an EMS (or a device) offers flexibility options at its interface to adapt its behaviour. They can be activated to trigger adapted power injections. They can also be aggregated for the next hierarchical level to provide flexibilities with higher volumes. *Therefore, the activation and aggregation of flexibility to restore a desired schedule is the scope of this case study from the coordination point of view.* We describe the details further below.

5.3.2 EMS Components in a Hierarchical VPP

In our VPP case study we extend the classical central VPP structure by introducing intermediate systems between the central system and the leafs. We call those intermediate systems *Cluster-Heads*. From the architecture perspective, a *ClusterHead* is a software component, which has the same core functionality as the hierarchically higher management systems, but is responsible only for a subset of its components. By dividing the central system into smaller subsystems, we establish a hierarchy with intermediate EMS that enable to hook up device EMS to different clusters. Particularly, the monitoring and control capabilities exist on the intermediate hierarchical levels as well. The original central system integrates a *ClusterHead* as a regular controllable component. The *ClusterHead*, however, abstracts from internal details and its internal composition. This is shown in Figure 5.2. Multiple intermediate *ClusterHead*-layers are possible with that structure. We can even create a cluster that consists of several smaller clusters, or re-arrange the composition of lower components. The benefit of the hierarchy is that the architecture (i.e. communication channels and behaviour) of the higher management systems and the involved applications do not change, since most of the work is already done at lower layers.

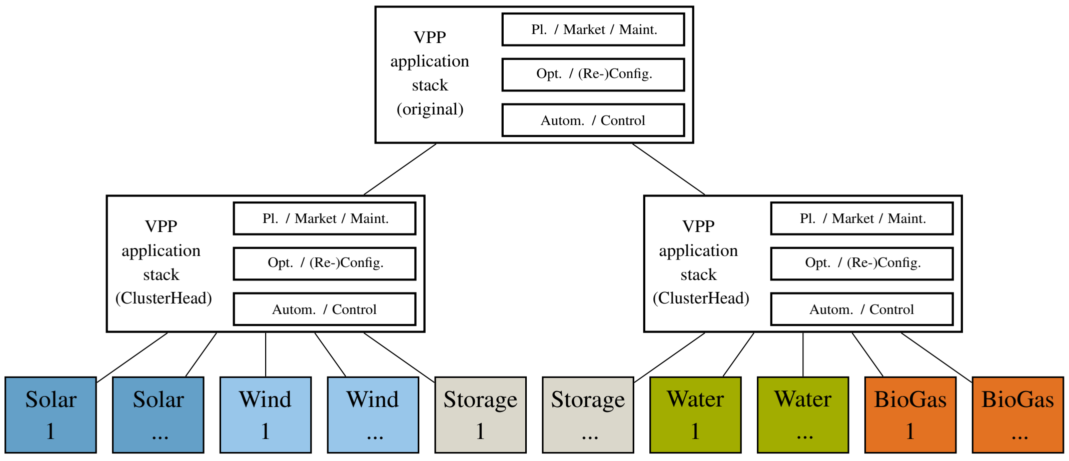


Figure 5.2: Hierarchical VPP architecture with intermediate components.

The hierarchical concept looks simple and clean, but the simplicity comes at the cost of additional specification effort. Particularly, a meaningful abstraction of EMS is required to create clear interfaces for the hierarchic architecture. We introduce those abstractions to describe the interface and explain how to work with them during our study execution.

5.3.3 Demonstrator Environment and Technical Constraints

To demonstrate the technical feasibility a technical demonstrator is part of the case study. It is designed to reproduce the technical environment of a real VPP. To be close to the real setting as possible, we performed several workshops with our industrial partners to determine the essential concepts and activities related to the integration of the components and to find out the involved coordination functionality.

During the workshops, the Munich utility presented their current system architecture, which we cannot depict here in detail due to the confidentiality reasons. However, the setting is similar to a conventional SCADA system. In general, the connected components are at remote locations, where they have the generation unit and a technical control system. The technical control system is often a classical PLC, such as the SPS-S7 from Siemens. To integrate it into a VPP system, it is connected to a so-called 'Fernwirkanlage' (FW), which is a hardware component with communication capabilities to establish a secure remote connection. Currently, the Munich utility uses the hardware of the type SAE-FW5 and SAE-FW50 with their own VPN system. For the data exchange the industrial standard IEC 60870-5-104 is used. The FW components act as gateways. In addition, they offer the possibility to buffer data and create histories as a backup solution, if communication is not available at some point of time. Furthermore, in terms of control, the FWs cannot provide a local schedule without additional software at the time of writing this thesis, but only react to set-points communicated by the VPP system. However, it is possible to extend the functionality of the FWs by another local device that has this capability and is locally connected to the FW. For instance, we can use a device similar to a Raspberry Pi (but which is more reliable) that has the additional functionality and connect it locally to the FW. This possibility serves as the extension point for the case study. We keep the whole structure of the real VPP, including the technical protocols and secured communications, and extend the local system with a new device that allows to add additional local functionality for communication and control. Later on, according to the process engineering department of the utility, the best case would be to replace these local devices with certified utility software plugins that are installed directly at the manufacturers control systems, so that no utility owned hardware is required at all.

Furthermore, in terms of the desired functionality, we discussed in another workshop the current operation schemes of VPPs in order to understand the necessary control options that we need to consider in our hierarchic VPP case study. According to the engineers, the VPP operation has two modes for each device, an autonomous mode and pool-operation mode. In the pool-operation mode, the central controller controls each generation unit. It means it can be turned on and off as well as controlled with a set-point that it must follow. In the autonomous mode, each generation unit does not follow a set-point, but rather its internal logic. For instance, CHPs operate temperature-controlled, while photovoltaic and wind produce as much as possible. However, even in the autonomous mode, the devices communicate their current values to the VPP. If communication breaks down, the devices activate automatically the autonomous mode and locally connected FW collects the history of production.

Introducing the intermediate *ClusterHeads*, as presented shortly before, intends to implement the same set of functionality as a VPP, i.e. to be seen as a VPP for the generation units and also the same set of functionality as generation units to be seen as a generation component by the higher level VPP. Hence, the *ClusterHead* must provide the following functionality:

- Monitoring of the devices and providing aggregated monitoring data for the VPP
- Control of devices to keep a desired schedule
- Providing aggregated control options for the next higher hierarchy level
- Local optimizations to create schedules for its devices (optional)

In summary, to execute the case study we need to consider the following technical conditions. First, we require hardware equipment that represents the generators and their controllers. Second, we require an interconnection of the FW components to show the integration into the SCADA system and the usage of industrial communication protocols. Third, we require to represent the VPP software functionality, in the sense of its control logic that is adapted to the hierarchical structure. This can be done with our presented SESSIM co-simulation framework that we presented in the last part of Chapter 4.

Note as a short disclaimer, the presented case study stays on a research prototype level. We cannot evaluate its technical feasibility in real running systems. The reason is that real VPPs are critical for the system stability and have a large economic impact. However, to plausibilize our results we use real hardware devices and open source SCADA systems that are used in industrial applications as well. They are specified and provided by our industrial partners Stadtwerke München (Munich Utility) and IBH Systems, the company behind the open source project *Eclipse neoSCADA*[1].

5.4 Study Execution

In this section we describe the steps that we performed in our case study based on the selected coordination goal and the presented technical environment of the study design. The general structure of our architecture as shown in Figure 5.2 serves as our guidance. In our first two steps we start with the basics by specifying the system interfaces and system behaviour of a VPP coordination system that considers only one hierarchical layer to work out the basic functionality. This specification is based on our approach presented in Chapter 4. While the first step only presents the abstract interfaces, the second step continues with a refinement of both the interface and the behaviour to describe how more advanced coordination system can be developed. In the third step, we extend the model to a multiple layer hierarchic system and study how coordination is achieved in hierarchical systems. Here, particularly the composite design pattern is of high interest, to extend the functionality and introduce additional considerations for aggregation and delegation. After the hierarchic coordination system was described mathematically, the fourth step shows how the system is implemented and how our architecture design principles together with the co-simulation framework helps to create real systems for further evaluations. We also show the usage of the demonstrator and outline its graphical user interface.

5.4.1 Step 1: Specification of the Interface and Behaviour

The central idea of a hierarchic architecture for EMS coordination is to specify an interface and behaviour of a system that processes input data from the adjacent hierarchical layers. This defines the necessary system boundaries to create a software based coordination system. Therefore, we need to understand what data shall be exposed to external systems an what data is kept within the local system. One of the most important aspects is that local configurations for EMS systems exist. Usually this is certain control logic that is desired locally. For instance, internal parameters exists to keep certain operational limits or achieve some kind of optimization for the local energy

[1]Eclipse neoSCADA: `www.eclipse.org/eclipsescada`, last access in March 2020

usage. This configuration is not intended to be overwritten or exposed to external applications, due to safety considerations and privacy issues. The first step is therefore to find EMS system functionality that can be exposed to external systems.

Our approach from Chapter 4 helps in the specification task with a set of atomic cases. The specification of those cases is shown in detail in Table 4.3. We show an abbreviated version in Table 5.1 for a better overview of the upcoming specification activities. We start with a first functionality for our case study and identify the right atomic case for that.

Case		Requires (next)
🟦	P1: delegation	→ 🟦
🟦	P2: direct answer (to parent)	→ ⬜
🟦	P3: state update (due to parent)	
⬜	C1: direct answer (to child)	→ 🟦
⬜	C2: support	→ 🟦
⬜	C3: escalation	→ ⬜
⬜	C4: coordination	→ 🟦
⬜	C5: aggregation	→ ⬜
⬜	C6: state update (due to child / children)	
🟧	L1: triggered message to child / children	→ 🟦
🟧	L2: triggered message to parent	→ ⬜
🟧	L3: state update (locally)	

Table 5.1: An abbreviated overview of the case distinction of EMS in a hierarchic architecture.

Monitoring

A common functionality for the (hierarchic) VPP is the monitoring of connected components. Connected components usually push their status to the VPP. This fits well to the **L2: triggered message to parent** case. Further, this (and every other case) involves data exchange between systems. Data is important to define the interface. The next step is to identify the required data that is being exchanged in the L2-case.

A VPP connected system produces electrical power. The current `PowerProduction` is therefore an important data point. It contains quantitative information about the power production (power injection). Note, for a VPP we do not need necessarily details like phases, phase angles, currents, etc. We assume that those values exist inside the EMS, but they are not exposed for the L2-case. The next data points is `ScheduledPower`. It reflects the desired operation at the given time point and considers expected maintenance shut down times. The schedule can be given in percentage of the installed capacity (e.g. photovoltaic generation) or with an absolute value.

Further, to keep track of additional statistical data, we also need static information about the components. First, the `ProductionCapacity` is required to understand the components maximum available power rating. Also the `ProductionType` of the power plant, i.e. weather the electrical power is produced from solar, wind, water, gas, coal, etc., is important to generate reports for energy identification and higher transparency in accordance to national laws (Germany: EnWG §42) and EU Directives. For composite components (ClusterHeads) the `ProductionType` represents the aggregated information about the subcomponents. Further data that describes components in more detail, like installation date, average availability, etc. are of course important for the management system in terms of administration and maintenance, but we neglect those data here for the sake of clearness of our case study.

To reflect the possibility for control a component can offer the possibility to increase and decrease its current production, depending on its type. Therefore the next useful information is denoted as available `Flexibility`. Flexibility consists of several sub-data points. Important ones are the available volume, the speed of its ramp up and ramp down capability and the price information to steer an economic driven optimization. The price, the ramping speed and the schedule allows optimized planning and control for the coordination. We will explain flexibility data later in more detail.

The data fields are used for the interface specification of the L2-case. They are summarized in Table 5.2, where we use the prefix `O:` to denote that these data belong to the output interface. The table represents the abstracted view on the output of each EMS towards the higher hierarchy levels. The abstracted view reduces complexity and provide information hiding to avoid the exposition of internal details. The data is used for monitoring as mentioned earlier, but it can also be used for further use cases like visualization or other.

Case	Input	Output

L2	none (e.g. time-based)	`O:PowerProduction`
		`O:ScheduledPower`
		`O:Flexibility`
		`O:ProductionCapacity`
		`O:ProductionType`

Table 5.2: The interface specification with the corresponding data fields for L2-case (triggered messages to upper layers).

With respect to the behaviour, this case is rather simple. A software behaviour is in general a mapping of its input to its output. Of course, it has also some internal states that influence its output. For monitoring, there is no dedicated input necessary. We assume that the output is generated only from internal states that are available at its interface, for instance based on defined time intervals, for instance every second.

After the definition of an initial atomic case, the next step is the definition of related cases according our approa<ch. We use Table 5.1 as reference, where the L2-case has follow up relation to further *C-case specifications*. It is not necessary to specify all cases, because their necessity depends on the application requirements. Nevertheless, the relation provides a good orientation how to systematically continue with sequent specification actions. The interface specification of the L2-case represent inputs for the C-cases.

In our hierarchic VPP case study, we are particularly interested in two further atomic case that might be triggered by the L2-case: aggregation to provide the same monitoring data to the next hierarchy and coordination that might happen within each individual cluster but also over different hierarchical levels. This involves the specification of the **C5: aggregation** and **C4: coordination** cases. We present the C4-case in the following and the C5-case further below, when we present multiple hierarchical levels.

Control Signals (Coordination)

While the aggregation prepares the data for the upper hierarchy, the coordination case (C4-case) considers the communication downwards, where a *ClusterHead* component collects the data from its children, processes the status and sends a message for the coordination back to its children, if some action is required. The children evaluate the request of the parent and transfer this signal into new set-points. The input / output interface of the C4-case is shown in Table 5.3 below.

Case	Input (n-channels)	Output (n-channels)

C4	`I:PowerProductions`
	`I:ScheduledPowers`
	`I:Flexibilitys`
	`I:ProductionCapacitys`
	`I:ProductionTypes`
		`O:RequestForAdaptations`

Table 5.3: The initial C4 case coordination, including the corresponding data fields at the interface.

In our case study, the coordination case is used for the automation of a cluster. A set of children send their status to the parent, which is a *ClusterHead*. The parent checks if the produced power and the desired schedule are as expected. If this the case, no adaptation is required and the automation request content is a signal that no adaptation is required. If a deviation is observed, the parent sends a request for its children, so that they change their power production to fit the schedule. We explain this in the following in more detail. Finally, the implementation of the control signal encoded in `RequestForAdaptation` corresponds to the P3-case (state update due to parent). This is simply updating its current set-point. The interface is straightforward.

5.4.2 Step 2: Handling of Flexibilities

After specifying an initial version of the interfaces, we approached in a second step the question how the flexibilities are used concretely for a coordination scenario. We split the question into two parts. First, we define how a direct implementation between a parent and its children is implemented (i.e. without the necessity for aggregation and delegation). In a follow up step (our third step), we investigated how this will be done over multiple hierarchies.

The major coordination goal in our case study is to maintain a desired schedule of a VPP. For that an EMS sends flexibilities to provide control capabilities of the power production to the upper hierarchy. A flexibility can be understand as an option that the component offers in order to adapt its production. It is positive, if the component is able to increase the production, and negative, if the component is able to decrease the production. For our case study, we use a number of attributes to quantify a flexibility. The attributes are:

- `Volume` (in kW) - it represents the maximum possible power adaptation of the component.
- `Ramp-Speed` (in kW/second) - it represents how fast a component reacts on desired increase or decrease of power.
- `Start-Delay` (in seconds) - it represents how long a component needs to show a first reaction, e.g. to turn on some generator, after is has received the signal to do so.
- `Price` (in EUR/kWh) - it represents the cost of the flexibility
- `Usage` (in kW) - it indicates how much of the flexibility is already being used.

For further tracking every flexibility has a time-stamp to describe when the flexibility was updated last time and an expiration time point that indicates how long the flexibility is valid to describe the temporal limitations that affect for instance batteries. The ClusterHead component uses the flexibility to adjust the power production, if a deviation from the schedule is observed. The adjustment is communicated over the request message, which defines the desired adaptation volume of the production (Figure 5.3).

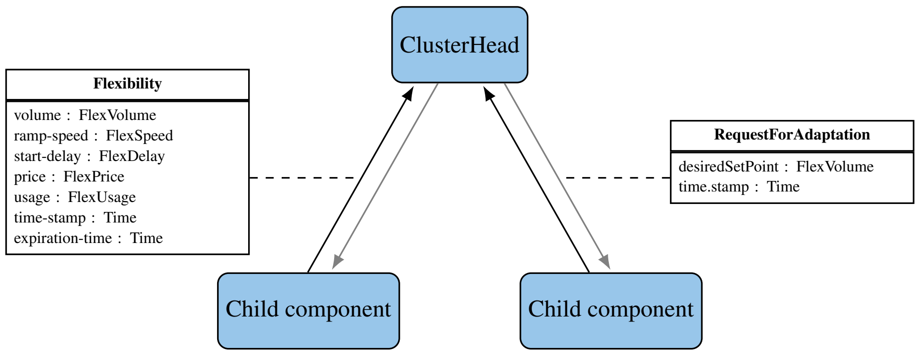
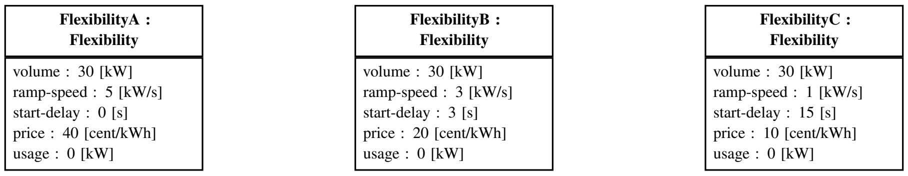


Figure 5.3: Monitoring and coordination with flexibilities.

To demonstrate the activation of the flexibility, we use an illustrative example. The example represents a coordination case with three different flexibilities.

Example: Coordination by activating available flexibilities

We assume that a cluster of EMS that is coordinated by one *ClusterHead* system (also an EMS). It is scheduled to constantly produce 100 kW of power. After 5 seconds of operation the production suddenly drops to 50 kW, for instance due to a failure of one component. The *ClusterHead* tries to restore the desired schedule. For that, it continuously collects the available flexibilities. We use three flexibilities in our example to explain the effect: `Flexibility A`, `Flexibility B` and `Flexibility C`. Each is different in its reaction speed, start delay, and price, but the maximum volume of the flexibilities is equal in this example. The values are shown in Figure 5.4. Further, we assume that there is a communication delay of one second. Additionally we use discrete time intervals of one second to demonstrate the behaviour. It means, that one second is required from the change of the status, until the children components receives the signal with new set points from the ClusterHead. The set-points are used to activate the flexibility according the specification.



FlexibilityA : Flexibility

volume : 30 [kW]
ramp-speed : 5 [kW/s]
start-delay : 0 [s]
price : 40 [cent/kWh]
usage : 0 [kW]

FlexibilityB : Flexibility

volume : 30 [kW]
ramp-speed : 3 [kW/s]
start-delay : 3 [s]
price : 20 [cent/kWh]
usage : 0 [kW]

FlexibilityC : Flexibility

volume : 30 [kW]
ramp-speed : 1 [kW/s]
start-delay : 15 [s]
price : 10 [cent/kWh]
usage : 0 [kW]

Figure 5.4: List of available flexibilities for our example.

In this setting we can establish the congruence of the schedule and current production in various ways (Figure 5.5). This is used to demonstrate, that even with the same interface various coordination mechanisms can be used. The simplest solution (see Figure 5.5 (a)) is the calculation of the difference between the schedule and the production. To reduce the deviation we use available flexibilities. If we activate the flexibilities only according the corresponding price, which means that we cover the missing 50 kW only with the cheapest flexibilities (`Flexibility C` and `Flexibility B`) we achieve a reaction is as shown Figure 5.5 (a). In our setting, the gap between the scheduled production and the observed production is closed after 51 seconds including the delays of the flexibilities, the ramp up speed and the communication delay.

The coordination algorithm can be improved (see also Figure 5.5 (b) and Figure 5.5 (c)). For instance, if we choose an approach that immediately activates all flexibilities, in order to close the gap as quickly as possible. After that the coordination algorithm monitors the current reached states. As soon as the desired level is reached, it prefers using the lower priced flexibilities instead of the more expensive ones. In this process it demands that cheaper flexibilities are ramped up, but it does not consider the communication delay. Therefore, the newly determined set-points reach the components with one second delay. This behaviour is demonstrated in Figure 5.5 (b). After the initial gap is closed, the ClusterHead replaces `Flexibility A` by `Flexibility B` and `Flexibility C` according their ramp up speeds. We see that in this regime the scheduled production is ramped up until 18 seconds. Then there is an overshoot due to delays in communication, which lead to the delayed updates when each flexibility reaches it scheduled level. The final state is established after 53 seconds, when the `Flexibility B` and `Flexibility C` have the desired values and the delay due to the communication vanishes.

This coordination algorithm can further be improved using a simple linear programming (LP) optimization model that takes all options into account (including all ramp up speeds and the delays) and optimizes them in one step that leads to schedule consisting of list of time stamps and related set-points for each flexibility (shown in Figure 5.5 (c)). The LP-model has the similar general form as for short time planning:

$$\min_{\mathbf{x}} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{x} \tag{5.1a}$$

$$\text{s.t.} \quad \mathbf{Ax} = \mathbf{b} \tag{5.1b}$$

$$\mathbf{Gx} \leq \mathbf{h} \tag{5.1c}$$

$$\mathbf{x_{lb}} \leq \mathbf{x} \leq \mathbf{x_{ub}}, \tag{5.1d}$$

where $\boldsymbol{\lambda}$ is the cost vector containing the flexibility prices, $\mathbf{x}$ the discretised vector of the flexibility power contribution and the missing power, $\mathbf{b}$ the power delivered by flexibilities, $\mathbf{A}$ the

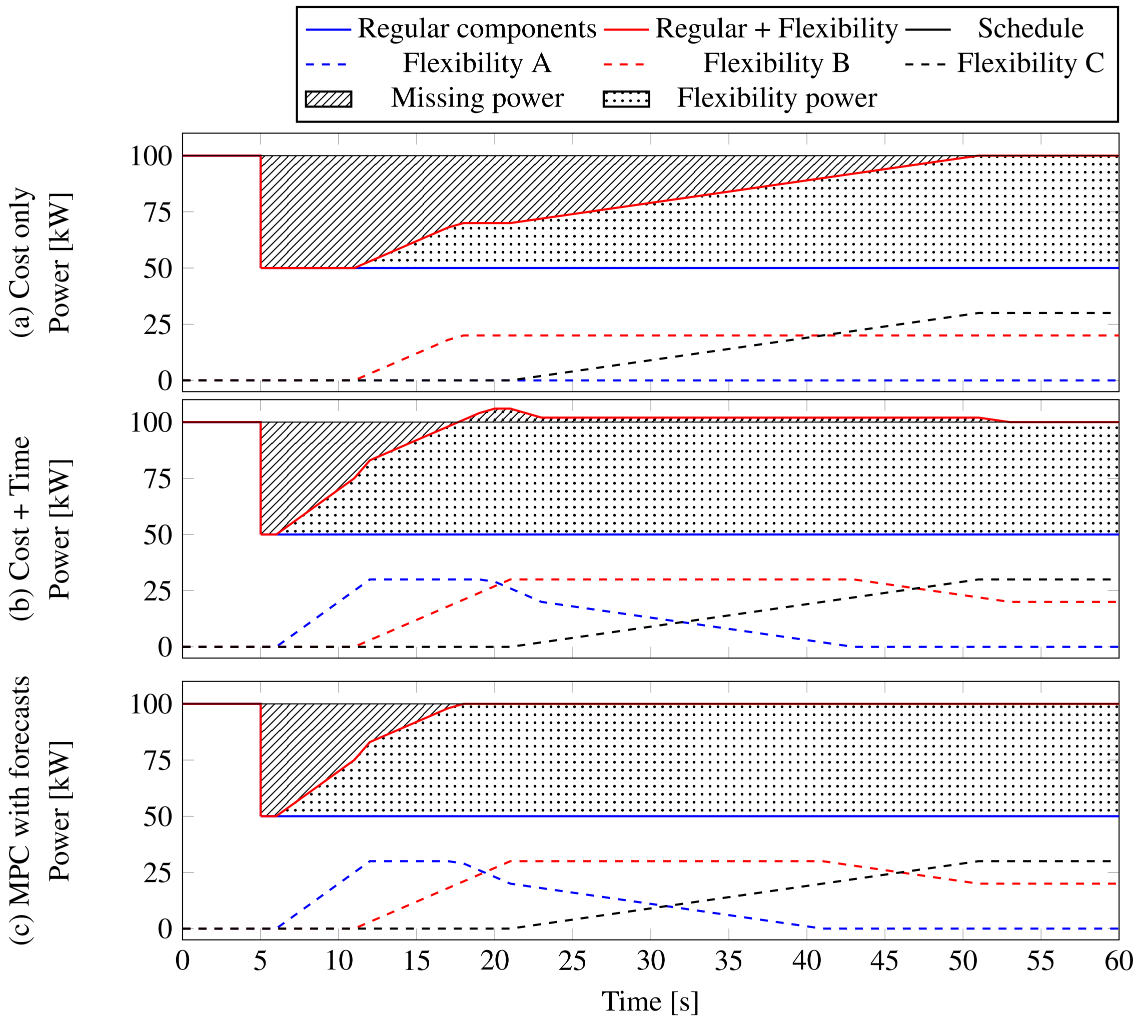


Figure 5.5: Different coordination approaches of using flexibilities for control.

coupling matrix, $\mathbf{G}$ the matrix for the temporal dependencies to consider ramps and delays, $\mathbf{h}$ the corresponding ramp and delay constraints, $\mathbf{x_{lb}}, \mathbf{x_{ub}}$ the volume constraints.

To create the matrices we use the following linear equations. First, we state the power conservation law for each time step k:

$$P^k = P^k_{mis} + \sum_{i=1}^{n} P^k_i, \tag{5.2}$$

where P^k is the deviation to the schedule that need to be closed by the flexibilities, P^k_{mis} is the missing power (see also Figure 5.5) and P^k_i is the power of flexibility i at time k with n the number of flexibilities, with $i, k, n \in \mathbb{N}$ and $P \in \mathbb{R}$. Further, the volume and the ramp-speed constrain the power activation for each time k as:

$$0 \leq P^k_i \leq P^k_{i,max}, \tag{5.3}$$

$$0 \leq |P^{k+1}_i - P^k_i| \leq \Delta P_{i,max}, \tag{5.4}$$

where $P^k_{i,max}$ represents the maximum power constraints at time k, i.e. it is zero during the start delay intervals and the available flexibility volume otherwise, while $\Delta P_{i,max}$ represents the maximum ramp speed of flexibility i between two time steps. Finally, by choosing the production vector as

$$\mathbf{x}^T = (P^1_{mis},\dots,P^H_{mis},P^1_1,\dots,P^H_1,\dots,P^1_N,\dots,P^H_N), \tag{5.5}$$

where H,N denotes the chosen optimization horizon and the number of available flexibilities, we can define $\mathbf{h}$ with the corresponding constraints due $\Delta P_{i,max}$ representing the ramp-speeds and the constraint vectors $\mathbf{x_{lb}}$ and $\mathbf{x_{ub}}$ as zero and $P^k_{i,max}$, respectively. This determines all necessary equality and inequality constraints in order to formulate a classical LP optimization problem.

The optimization formulation for the ClusterHead allows to calculate the required flexibilities with a model predictive controller (MPC) by applying the received information at hand. Instead of single set points, the ClusterHead sends a list of set points, with the desired time stamp, so that the individual flexibilities are controlled more precisely. This removes the overshoots and the effect due to the communication delay. The MPC based behaviour is shown in Figure 5.5 (c). With this setting, the gap is completely closed after 18 seconds, and the remaining flexibility replacement is achieved more smoothly.

Finally, the coordination process leads to different costs for the activation of flexibilities at hand. To compare the costs, we make one last additional assumption. First, we use the cost parameter of each flexibility. Further, we consider the deviation of power from the schedule is a non-desired state of the system. We therefore punish this state and assume for that deviation a cost of 100 cent/kWh. With this, we easily calculate the costs of each algorithm, denoted with C_{total} as an integral over time,

$$C_i = \int_{t_{start}}^{t_{end}} C_i \cdot |P_i(t)|\, dt \quad \text{and} \quad C_{total} = \sum_{i=0}^{n} C_i, \tag{5.6}$$

where C_i is the individual cost of a component, and $|P_i(t)|$ the absolute value of the power. In our example, we have $i \in [0..3]$, where $i = 0$ represent the deviation from the schedule, hence C_{mis} and $i = 1,2,3$ the activation of the flexibilities. With that, our costs are calculated as shown in Table 5.4.

Coordination approach	C_{total}	C_{mis}	$C_{Flexibility_A}$	$C_{Flexibility_B}$	$C_{Flexibility_C}$
Figure 5.5 (a)	38.8 cent	31.7 cent	0.0 cent	5.1 cent	2.0 cent
Figure 5.5 (b)	26.8 cent	11.4 cent	6.7 cent	6.7 cent	2.0 cent
Figure 5.5 (c)	23.8 cent	9.2 cent	6.0 cent	6.6 cent	2.0 cent

Table 5.4: Cost comparison for different coordination approaches with flexibilities.

The example shows that the coordination with flexibilities can be realized only with data from the specified interfaces from the L2-case and the C4-case and that there can be different coordination approaches that lead to different results, depending on the choice of the algorithm.

5.4.3 Step 3: Flexibilities in Hierarchic Systems

The example above describes the coordination with only one hierarchical layer. Its extension to multiple layers is therefore the next step in our case study to finish the coordination and aggregation activity. The desired hierarchical structure is presented in Figure 5.6. We continue with the specification for the **C5: aggregation** case, with a mapping of the input to the output data at the corresponding interfaces. Note the input and output interface supports the same data types (see also Chapter 3). This case leads to the composition design pattern, which is particularly important, if we are interested to create a system that consist of similar functionalities at the different hierarchical levels.

Monitoring part 2 (aggregation)

The functionality for aggregation involves the collection of data from the children, their aggregation and the provision of aggregated data to the upper levels. The syntactic input to output relation, thus, the system boundary of the software system as presented in Section 3.2 is shown below in Table 5.5.

Case	Input (n-channels)	Output
C5	`I:PowerProductions`	`O:PowerProduction`
	`I:ScheduledPowers`	`O:ScheduledPower`
	`I:Flexibilitys`	`O:Flexibilitys`
	`I:ProductionCapacitys`	`O:ProductionCapacity`
	`I:ProductionTypes`	`O:ProductionTypes`

Table 5.5: The initial C5-case aggregation, including the corresponding data fields at the interface.

For EMS coordination, there are potentially several hierarchy layers, in which each EMS might coordinate a number of other EMS components. Current VPPs handle several hundreds, sometimes even more than one thousand of components and the number of connected components is growing. If we assume that each ClusterHead handles around 20 children, its hierarchically higher component will already handle 20^2 components, and the next layer 20^3, and the next on 20^4, which is already more than one hundred thousand components. Scalability is therefore important. In consequence, we cannot handle so many flexibilities with an MPC and need to reduce the number of data points to coordinate the components.

One ClusterHead maintains several children and acts as a child to the next hierarchical level. It aggregates the state of its children, the schedules and the available flexibilities. The aggregation behaviour (the mapping of the inputs values to the output values) is straightforward for almost all data points. Mostly, the aggregated value is a simple sum of the individual inputs. For instance, the `PowerProduction` in a VPP is a sum of its children, since losses are not considered in a VPP (this is the task of the transmission and distribution system operators). The aggregated `ScheduledPower` and the `ProductionCapacity` as a static value are also a sum of their constituents. The `ProductionType` is a weighted composition, based on the types of the

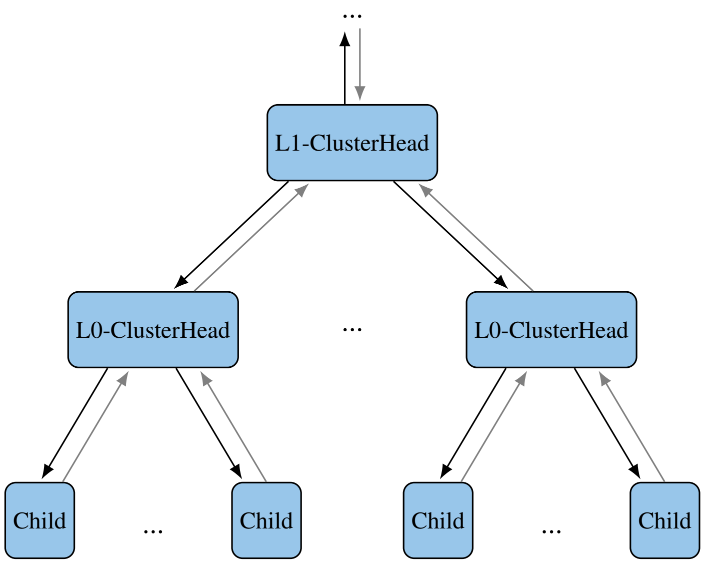


Figure 5.6: Hierarchy with multiple layers.

children and their power production. It describes how much energy[2] of which type is produced by the children. Flexibilities are aggregated and processed not as a sum. Their aggregation is more complex, because we try to avoid sending large lists of all available flexibilities that are collected upwards, due to scalability.

In our case study we used a combination of individual flexibilities to create flexibilities with a larger volume during the aggregation process. Similarly, as for coordination, potentially different aggregation algorithms might exist. For the basic understand about aggregation, we describe only one example that was done in our case study.

The aggregation is based on two steps. We start by sorting the flexibilities based on their price. Then we create several price regions to create multiple lists of flexibilities in a certain price range. In the second step, we aggregate of the flexibilities of one price region to generate a one single flexibility. We use the same ClusterHead systems from the previous example to explain that behaviour.

Example: Aggregation of Flexibilities

Assume that two ClusterHeads have received three flexibilities, each. The flexibilities are similar as before, but to see an effect we slightly vary the parameters of the previous example. This is presented in Figure 5.7. The modifications are shown in red. The ClusterHeads are marked with a number to identify their flexibilities.

Both ClusterHeads aggregate the flexibilities before communicating them upwards as following. They receive three flexibilities each with a price range is $[10..40]$. To keep the example easy to understand, the ClusterHeads create three price ranges, which are $[10..20), [20..30), [30..40]$. The first step is the grouping of the flexibilities according the price, which is straightforward in this example.

[2]Statistics consider time intervals of the type.

Flexibility 1A
volume : 30 [kW]
ramp-speed : 5 [kW/s]
start-delay : 0 [s]
price : 40 [cent/kWh]
usage : 0 [kW]

Flexibility 1B
volume : 30 [kW]
ramp-speed : 2 [kW/s]
start-delay : 3 [s]
price : 12 [cent/kWh]
usage : 0 [kW]

Flexibility 1C
volume : 30 [kW]
ramp-speed : 1 [kW/s]
start-delay : 10 [s]
price : 10 [cent/kWh]
usage : 0 [kW]

Flexibility 2A
volume : 70 [kW]
ramp-speed : 5 [kW/s]
start-delay : 0 [s]
price : 40 [cent/kWh]
usage : 0 [kW]

Flexibility 2B
volume : 30 [kW]
ramp-speed : 3 [kW/s]
start-delay : 3 [s]
price : 35 [cent/kWh]
usage : 0 [kW]

Flexibility 2C
volume : 30 [kW]
ramp-speed : 1 [kW/s]
start-delay : 10 [s]
price : 10 [cent/kWh]
usage : 0 [kW]

Figure 5.7: List of available flexibilities in a hierarchic system with two ClusterHeads.

The second step is the aggregation of all flexibilities within one price range. Here, this affects the flexibilities 1B with 1C, and 2A with 2B. For a proper aggregation, we use another representation for the flexibilities, which is aggregated more easily. Instead of the data fields that we got from the first example, we present a flexibility as time series graph. The time series describes the behaviour in detail, in particular it has the information about the ramp speeds and the delays, which are important for the MPC coordination. The aggregation is achieved with a superposition of the time series graph, as illustrated in Figure 5.8. The price is calculated as a weighted average using the volume of the flexibilities. For instance, the aggregated price of 1B+1C is 11 cents and 38.5 cents of 2A+2B.

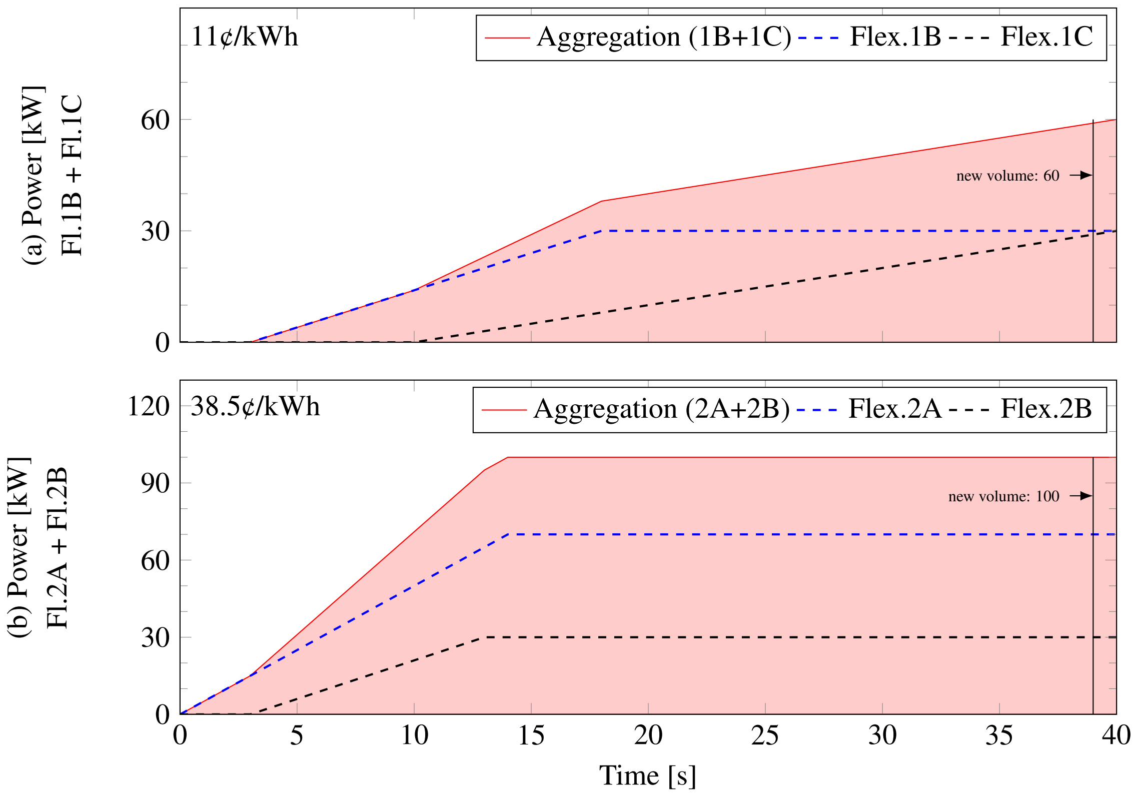


Figure 5.8: Aggregation of flexibilities.

The aggregation with a time series graph is simple and accurate for the technical operation. This is important for the coordination process. Note, however, that the price information becomes an approximation, due to the delays and ramp speeds. After the aggregation, the flexibilities are communicated upwards as messages, which are shown in Figure 5.9.

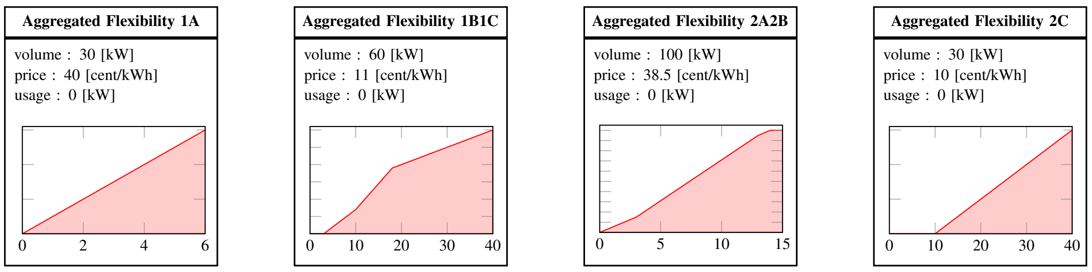


Figure 5.9: List of aggregated flexibilities to the next level.

The next level EMS can aggregate with the same mechanisms. In our example, from the four received flexibilities, the ClusterHead is able to create two flexibilities, where one of them has the volume of 130 kW and the price of 38.85 cents/kWh, and another one has the volume of 90 kWh and the price 10.67 cents/kWh. The ramp speeds and delays can be calculated through the graphs as demonstrated before.

The benefit of this aggregation approach is twofold. A flexibility with a time series representation is great for control and optimization. It is particularly good for control techniques such as MPC. In addition, ideally, instead of models, real measurements of local controllers can be used to generate the flexibility graph. This would reduce the modelling effort and increase the confidence in a reliable system. The second benefit is that the coordination process remains scalable. The flexibilities are selected and processed as demonstrated before (Figure 5.5).

Control signals part 2 (delegation)

Finally, we expect that somewhere on top, a ClusterHead requests the aggregated flexibility from one intermediate ClusterHead. This leads to the final specification case **P1: delegation**, where the input is an updated schedule in form of an `RequestForAdaptation`. The output has the same data type as well. This requires the intermediate ClusterHead to decompose the flexibility into individual flexibility responses to its children. We specify the input/output relation as shown in Table 5.6, and explain its meaning with another example, where we show how delegation and disaggregation works with multiple layers.

Case	Input	Output
P1	`I:RequestForAdaptations`	`O:RequestForAdaptations`

Table 5.6: The P1-case delegation, including the corresponding data fields at the interface.

Example: Flexibilities in multiple-layer systems.

We assume that we have two clusters that are arranged as shown in Figure 5.6. Both ClusterHeads are responsible to balance 100 kW of load, each. The ClusterHeads have the flexibilities as shown before. To see how the flexibilities are used, we assume that after some time the load of the first ClusterHead suddenly increases by further 100 kW.

To counter the increasing demand of power ClusterHead-1 immediately activates all of its flexibilities, as described before using the MPC coordination approach. This is the first coordination step (the case C4). At the same time, it communicates its new status upwards together with the aggregated flexibilities (the C5-case). The ClusterHead above notices, that the system under him lacks 100 kW of production. Therefore, it starts with the coordination as well. Based on the four received flexibilities (see Figure 5.9) it calculates a better solution for the coordination. Also for this we use the presented MPC approach. The answer is then a message that includes the MPC set points as a response. The response message content is shown in Figure 5.10. The green graph represent the desired set-points. The red graph indicates the available flexibility reserves. This is the C4-case with hierarchical layers.

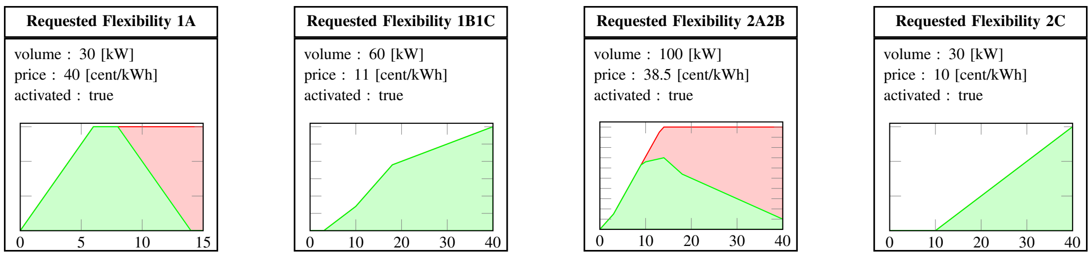


Figure 5.10: List of MPC requests for activation of flexibilities (to the intermediate ClusterHeads).

Further, the responsible ClusterHead disaggregates the coordination request. We show an economical way of disaggregation. The more economic solution is preferred and ClusterHead-2 chooses to activate flexibility 2B before 2A (Figure 5.11). This is the P1-delegation-case. Note, the response is the same as in the C4-case. Thus, the composite pattern effects the parents in a hierarchy as well. This is a slight variation compared to the object-oriented composite pattern.

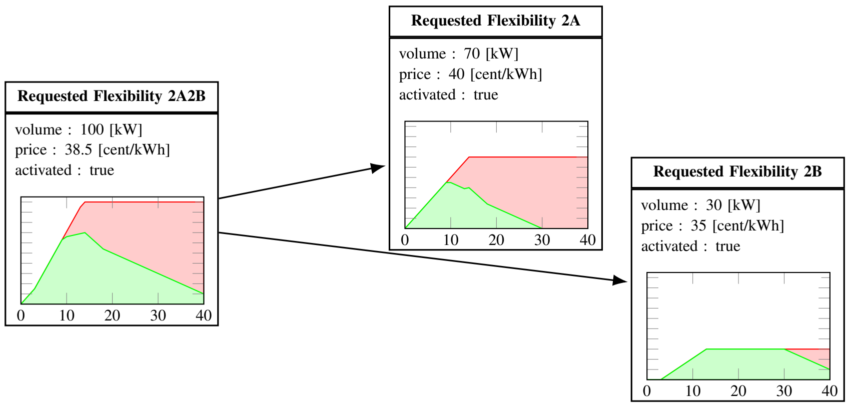


Figure 5.11: Disaggregation example of flexibilities.

Finally, the leafs react to the responses by activating the received flexibility set-points. This is the P3-case, a state update due to a parent's request. After that, no follow-up action is required, since monitoring is given by L2. We present the specification of the P3 interfaces below.

Case	Input	Output
P3	`I:RequestForAdaptations`	none application specific, however, an acknowledgement helps

Table 5.7: The P3 state update case due to parent's request.

The result of the behaviour is illustrated in Figure 5.12. The hierarchic system closes the gap of the 100 kW power production only 11 seconds after the load increases. Note, that the activation of flexibilities from ClusterHead-2 have an additional delay compared to ClusterHead-1. This is due to the additional hierarchy layer. This delay is a drawback on the one side, but on the other side, the hierarchic system increases is scalability if it activates the flexibilities of a group first and uses external flexibilities to improve the economic operation later on. This increases safety and keeps as much 'intelligence' as possible at the lower layers reducing potential coordination necessity.

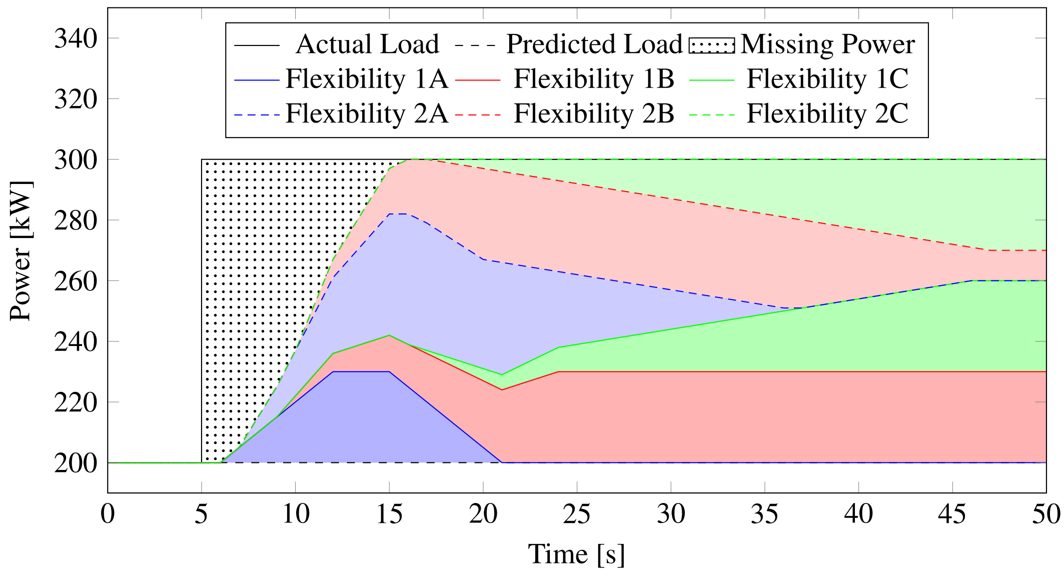


Figure 5.12: Balancing with flexibilities based on an MPC with aggregation and delegation.

5.4.4 Step 4: Architecture Design Patterns and our Coordination Architecture

Our first three steps show how a hierarchic coordination in VPP is designed from a functional perspective. Therefore, we have defined the system boundaries of the involved systems, which are from the software engineering perspective the system's interfaces. We also explained the behaviour for coordinating distributed resources of a VPP using flexibilities. We demonstrated only the coordination approach, but no real quantification for the power system operation. The reason is that the project scope was on designing the right interfaces with the hierarchic architecture[3]. In this step, we continue with the architecture design. We utilize the architecture design pattern that we presented in Chapter 4 to create a hierarchic architecture. We consider the following patterns:

(i) *client-server* - to cover remote interactions,

(ii) *composite pattern* - to ensure that each component offers the same interface data,

(iii) *layers* - to ensure that privacy, local safety constraints and security are respected,

(iv) *unified interfaces* - to ensure that each interface follows a unified representation of data and

(v) *publish-subscribe* (optional) - to allow better automation, by ensuring that data is only communicated when changes occur.

The architecture design patterns help us to achieve several construction challenges for the intended system architecture as initially presented in our study design in Figure 5.2. In a VPP, we have generation devices, i.e. hardware components with sensors and actuators; EMS that integrate the hardware components and offer an interface to external systems; Coordination components like ClusterHeads that integrate EMS and a larger system like SCADA. This leads to different layers as shown in Figure 5.13.

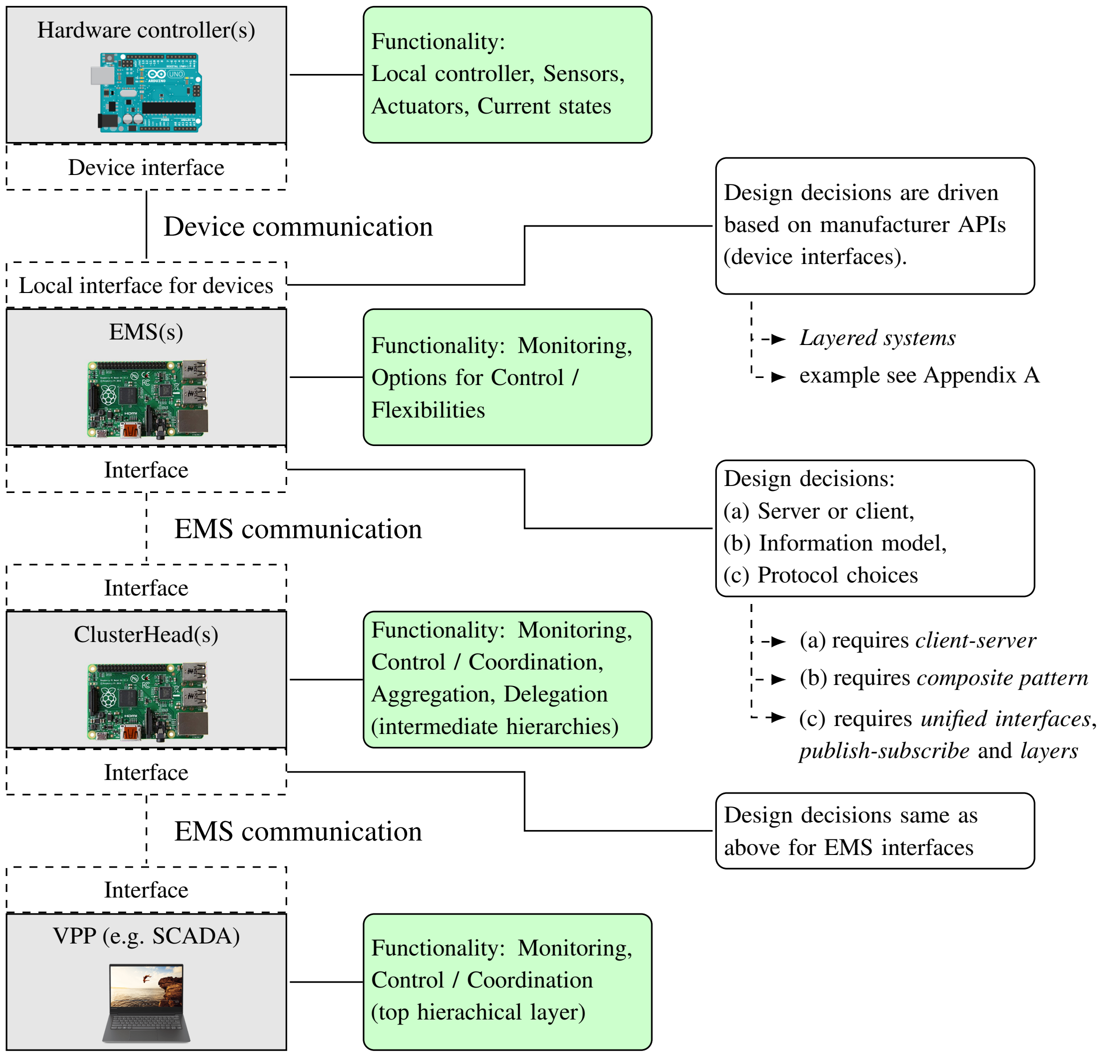
With such a system in mind, the architecture design patterns help in the following. First, the *client-server* demands us to choose, if the EMS that controls a hardware device is implemented as a *server* or a *client* towards the ClusterHead. Both is possible. A *server* has the advantage that it is reachable for external components. Hence, a server offers data that can be read from ClusterHeads (i.e. if the security constraints allow that). The consequence is that ClusterHeads collect the data using clients from their child EMS and offer their data again as another server to the next higher level. If we choose that the EMS exchanges its data as a *client*, the advantage is that it does not need to have be reachable by the public infrastructure. But on the other side, the ClusterHead has no possibility to request for data updates, but needs to wait, until the EMS sends it a new request. Therefore, we decided for an EMS interface towards the higher level to be implemented as a server, not a client in our case study.

Second, the *composite pattern* demands us to define the necessary data for the communication and a behaviour how this data is treated. This was already done in step 1 to step 3.

Third, the *layers* demand us to define whether some of the functionality can only be accessed through other layers. In our case study, we have decided that the interface of the EMS is only the first accessible layer. It can be strengthen by additional security measures with authentication, authorization, or encryption, like VPNs. However, the internal hardware lies behind the first layer. It can only be accessed from the internal functions of an EMS, not from external systems. We use this encapsulation to convert signals from the devices into the data defined by the interface. Layers are used similarly in more complex EMS as presented in one of our EMS demonstrators that is also introduced in Appendix A.

Fourth, the *unified interfaces* demand us to define which kind of protocol is used, but also which king of representation are available at the interface. We have chosen the widely

[3]Note, we show in Chapter 6 another case study including quantifiable results for the EMS coordination.



Hardware controller(s)
Functionality: Local controller, Sensors, Actuators, Current states
Device interface
Device communication
Design decisions are driven based on manufacturer APIs (device interfaces).
Local interface for devices
EMS(s)
Functionality: Monitoring, Options for Control / Flexibilities
Layered systems
example see Appendix A
Interface
EMS communication
Design decisions: (a) Server or client, (b) Information model, (c) Protocol choices
Interface
ClusterHead(s)
Functionality: Monitoring, Control / Coordination, Aggregation, Delegation (intermediate hierarchies)
(a) requires *client-server*
(b) requires *composite pattern*
(c) requires *unified interfaces*, *publish-subscribe* and *layers*
Interface
EMS communication
Design decisions same as above for EMS interfaces
Interface
VPP (e.g. SCADA)
Functionality: Monitoring, Control / Coordination (top hierachical layer)

Figure 5.13: Hierarchic architecture based on the presented architecture design patterns.

used REST (Fielding, 2000) interface with a JSON (Java Script Object Notation) Bray (2017) representation for the data. The reason was that the representation is simply converted from our Co-Simulation framework into that format and many libraries are available for the handling of that format.

Finally, the fifth and optional *publish-subscribe* architecture design pattern is necessary for some particular protocols. It also helps to improve the performance for certain use cases. However, we have not used it in this first case study, due to the previous design decision of the REST interface. Therefore, our servers show always the current values, but the clients have to actively ask for new updates. A subscription is not possible. We refer to the next Chapter 6 to demonstrate the implementation is a larger demonstrator to improve the operation.

5.4.5 Step 5: Implementation and Demonstrator

The presented case study was implemented in form of a demonstrator table and shown at the science fair *Münchner Wissenschaftstage* and the innovation fair of the Munich utility. The hardware is shown in Figure 5.14. The software perspective is presented in Figure 5.15. The demonstrator shows a number of components that represent the distributed hardware devices. They are integrated into distributed EMS (software components), which are integrated into a larger VPP structure using our co-simulation framework, which is introduced in Section 4.3. The demonstrator includes several components, which we explain in the following.

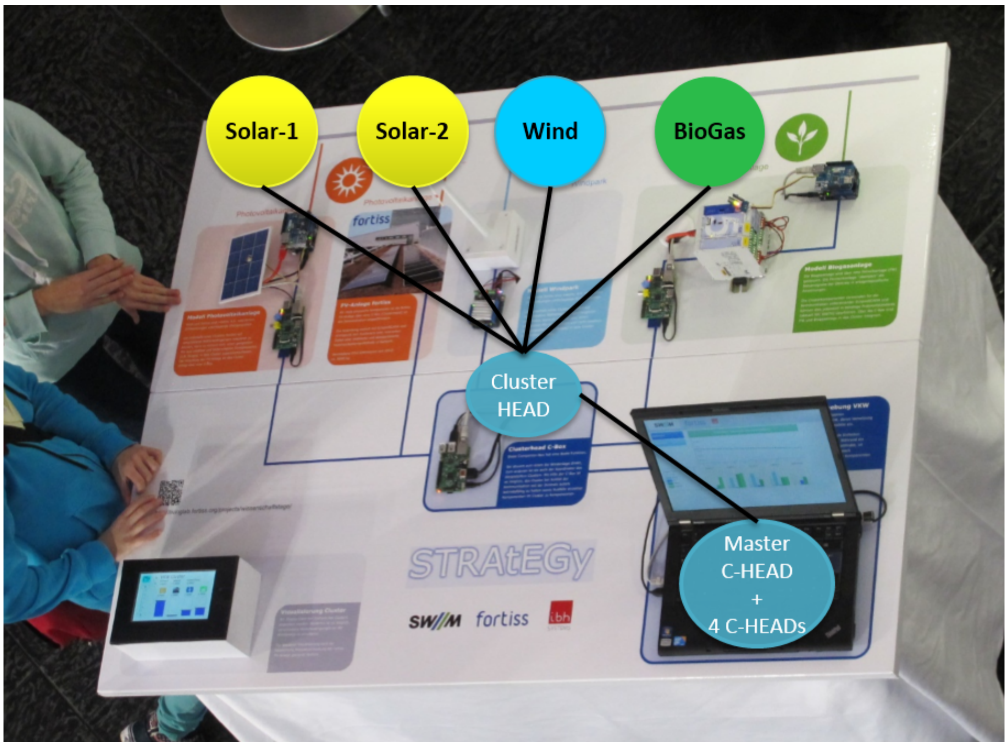


Figure 5.14: Demonstrator with real devices integrated with EclipseSCADA and Sessim.

For the field level, we have selected a number of embedded components. Those are a light sensitive photodiode that represents solar panels (Solar-1), a controllable motor mounted on a windmill model that represents a wind power generator (Wind), and different led stripes to show a status of the current set values for a biogas powered CHP (BioGas). Each component is connected to an Arduino board that is responsible for their control. The controller provides a variety of voltage levels for analogue control as well as digital I/O interfaces. The task of the controller is to provide the actual device data and react to set-points. To have a proper reaction of the components, we additionally implemented delays, ramp-up and ramp-down curves to see how the system deals with such delays and inertia of physical components. Arduinos are a good representation of field devices that read and set values for the operation of devices. In addition, we have a real PLC component as a demonstrator component, the net-line FW-5 micro telecontrol station that is provided from our utility partner SWM. It is used to validate that our approach can be combined with the IEC 60870-5-104 protocol.

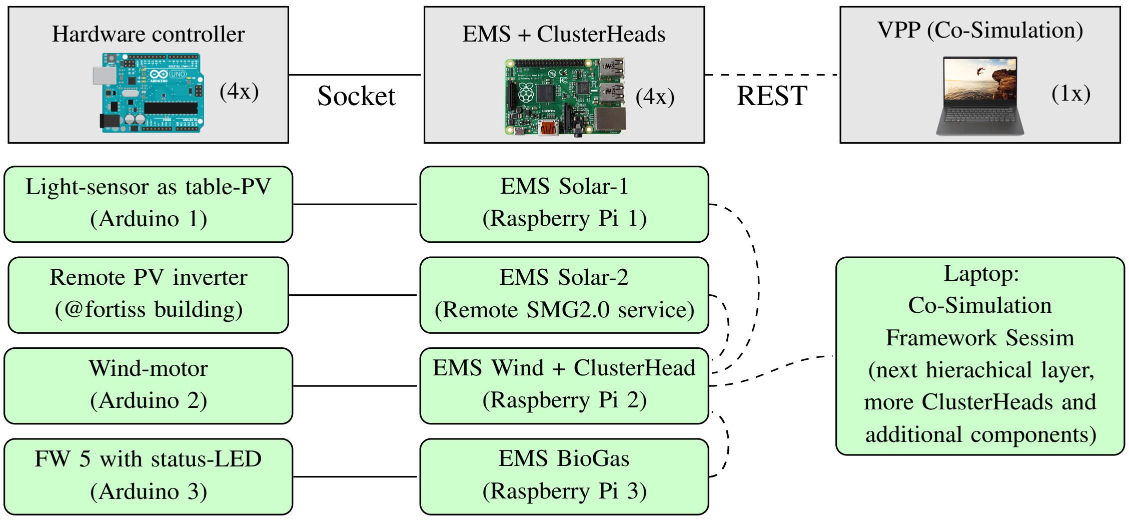

Hardware controller
(4x)
Socket
EMS + ClusterHeads
(4x)
REST
VPP (Co-Simulation)
(1x)
Light-sensor as table-PV (Arduino 1)
EMS Solar-1 (Raspberry Pi 1)
Remote PV inverter (@fortiss building)
EMS Solar-2 (Remote SMG2.0 service)
Wind-motor (Arduino 2)
EMS Wind + ClusterHead (Raspberry Pi 2)
FW 5 with status-LED (Arduino 3)
EMS BioGas (Raspberry Pi 3)
Laptop: Co-Simulation Framework Sessim (next hierachical layer, more ClusterHeads and additional components)

Figure 5.15: Deployment of our software components on the demonstrator hardware.

On the higher level, we have a number of RaspberryPis that represent local EMS systems. Each RaspberryPi establishes a connection to an Arduino over Ethernet using a socket connection. For the connection to the real PLC component, we use the Eclipse SCADA stack, which allows us to start a server on the RaspberryPi and collect the data from the PLC over the IEC 870-5-104 protocol. The connections are used to read the sensor values and write new set points for the actuators. The collected values are processed and exposed from the RaspberryPis with a REST-interface to any component that has access to the same network. They serve for the interaction with the ClusterHead components. The reason for choosing a RasperryPi was to demonstrate the distributed nature of the system. Further, a RaspberryPi has more processing power as an Arduino and much more flexibility for programming due to an operational system. It allows to run JAVA, in which we implemented the EMS, as well as to use a rich set of available libraries. Further, the RaspberryPis provide the possibility for remote access via SSH to update or deploy our software applications.

The next demonstrator software component is the ClusterHead. The ClusterHead software is also implemented with Java using Maven and Eclipse and runs on a RaspberryPi as well. For the implementation a variety of different libraries for the communication, visualization and the optimization algorithms were used. We deploy and start the ClusterHead software on one the EMS RaspberryPis to show that a coordination system can be deployed an any device.

The ClusterHead combines two major areas of functionalities, a monitoring and control of its cluster and its interaction with the higher hierarchic level. For the local cluster, the ClusterHead monitors and coordinates its components such that a desired schedule is preserved. Therefore, it periodically reads values from the EMS servers to calculate the current state and compare it with the schedule. In addition, it calculates the available cluster flexibility, from the component's capacities and their current state. Flexibilities are obtained as follows: Controllable devices, such as biomass generators or co-generation devices provide flexibility based on current production, their max/min limits with the consideration of ramps. For solar and wind devices we can get flexibility by adjusting the power point tracker or the wind blade angles, respectively. Hence, we get only the option to reduce the power as flexibility. For economical operation, flexibility options have a price tag. Next, if the ClusterHead observes a difference of production to the schedule, a control signal is generated with the MPC approach as described before. The signal is sent to the EMS servers to adapt their production.

It might happen that the difference to the schedule is larger as the available flexibilities. In that case, the ClusterHead's interaction with the higher hierarchic level becomes relevant. For the interaction the ClusterHead calculates aggregated values of the production and all available flexibilities. It provides them as a server over a REST-interface to the next higher level. The higher level ClusterHead is implemented with our co-simulation framework SESSIM. SESSIM runs on a conventional laptop. It provides not only a higher ClusterHead, but also more additional hierarchies for the VPP, further additional simulated components and a visualization. SESSIM, or more precisely, the ClusterHead parent actor within SESSIM reads the provided values from the ClusterHead on the table using its REST interface and simulates additional components that are clustered in groups[4]. The parent ClusterHead uses the information for further processing. This particularly includes the calculation of MPC set points, when the ClusterHead on a lower level is not able to cover the difference. The higher level ClusterHead has access to the flexibility of other ClusterHeads. It activates them as described before to keep the overall schedule of VPP. This allows that unexpected situations, such as faults or missing flexibility, are handled within the hierarchy, as presented before.

Finally, to show the interoperability with real systems, one additional RaspberryPi (solar-2) is directly connected to a photovoltaic system on the roof of our research institute fortiss over a mobile connection. For that we specifically created an interface at our SMG 2.0 system (introduced in Appendix A). It gets the data directly from the real inverter and provides the required data over REST to external components. The RaspberryPi on the table converts the data into the required self-similar hierarchic interface and interacts with the whole demonstration system exactly the same way, as with embedded devices. Hence, we show that the demonstrator covers a variety of embedded solutions, remote components and an extensive co-simulation environment with its hierarchic architecture.

Human Machine Interfaces

To demonstrate the effect for the coordination of EMS to a broader public we developed several graphical human machine interfaces (HMI) to show the functionality and the hierarchy. One HMI is established for the ClusterHead of the table (bottom left in Figure 5.14 and a larger illustration in Figure 5.16). Another HMI represents the higher level ClusterHead that represents the overall VPP system (bottom right in Figure 5.14 and a larger illustration of it is shown Figure 5.17 and Figure 5.18).

The HMI for the ClusterHead (see Figure 5.16) is used for the monitoring and automation of the devices on the table. It allows the coordination of local components and show how a balance is established automatically for the power production and the desired control set point.

[4]Scalability tests of SESSIM shows a simulation capability up to tens of thousands additional components, which clearly outperforms other agent based approaches such as JADE.

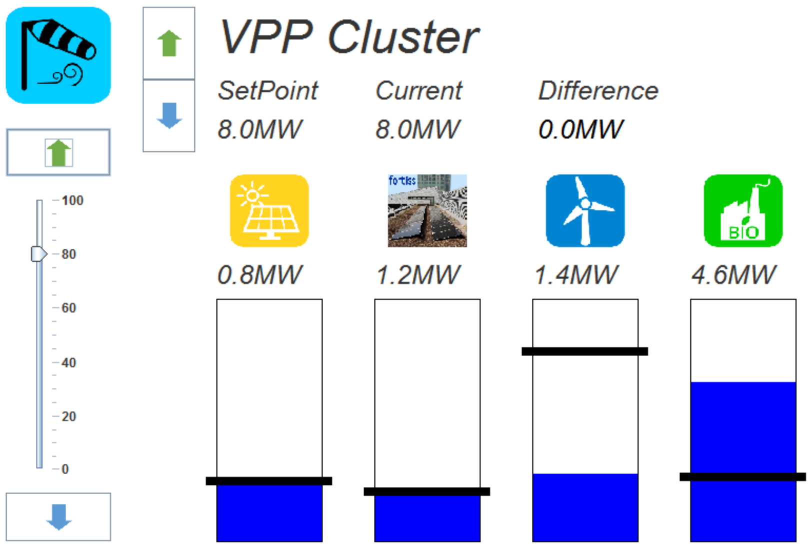


Figure 5.16: Interaction with the ClusterHead.

To experience the system's behaviour the ClusterHead's set points are varied with the panel using the arrows next to the label 'SetPoint'. All components are monitored and visualized on the panel[5]. The solar component on the left is the readout from the photosensitive sensor on the table. Increased lights or shading immediately influences the values. The second solar component is the readout from the remote PV-installation at the fortiss building. The third component is the readout from the wind generator motor. We have additional arrows on the left to adjust the 'wind' power in percent, in order to vary this component. Because the windmill has delays, directly after an adjustment the wind production is not the same as the desired value. This is shown by the deviation of the blue bar (current value) from the black cross bar (set point). The controllable fourth component is controlled using an MPC to establish the right balance.

The HMI of the VPP system, which runs on the laptop, shows how the ClusterHead interacts with higher level EMS. It has two views, the current production overview including the contribution of individual ClusterHeads (Figure 5.17) and the topology of the whole system that is available in the simulation (Figure 5.18).

The total VPP system consists of around ninety production components clustered in five groups. They are modelled according the data provided by SWM and a normalized set of data of real measured values that represents the schedule of the VPP. The allocation of the components to the ClusterHeads is chosen such, that each ClusterHead covers multiple types of components. Each component offers a flexibility according to its type. Solar and wind components offer negative flexibility meaning that they can reduce its power output. Water cannot be adapted (due to environmental regulations in Munich). Biogas and biomass components offer positive and negative flexibility based on their current production and their installed capacity constraints.

[5]Data values are scaled for demonstration purposes

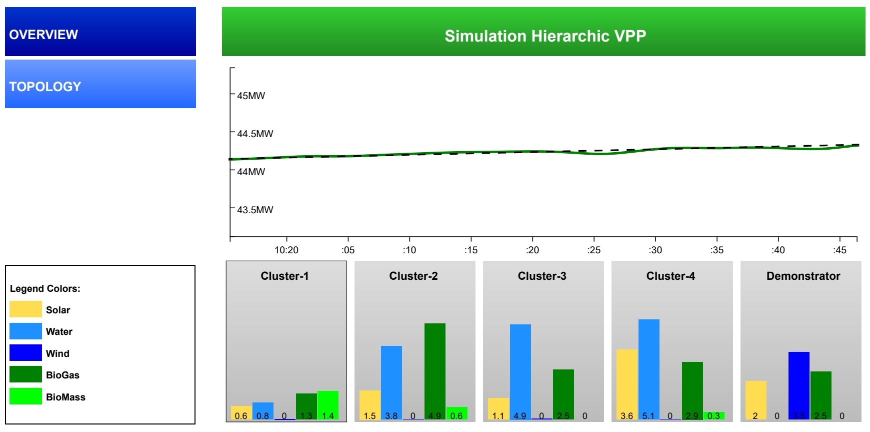


Figure 5.17: Higher level EMS that interacts with multiple ClusterHeads.

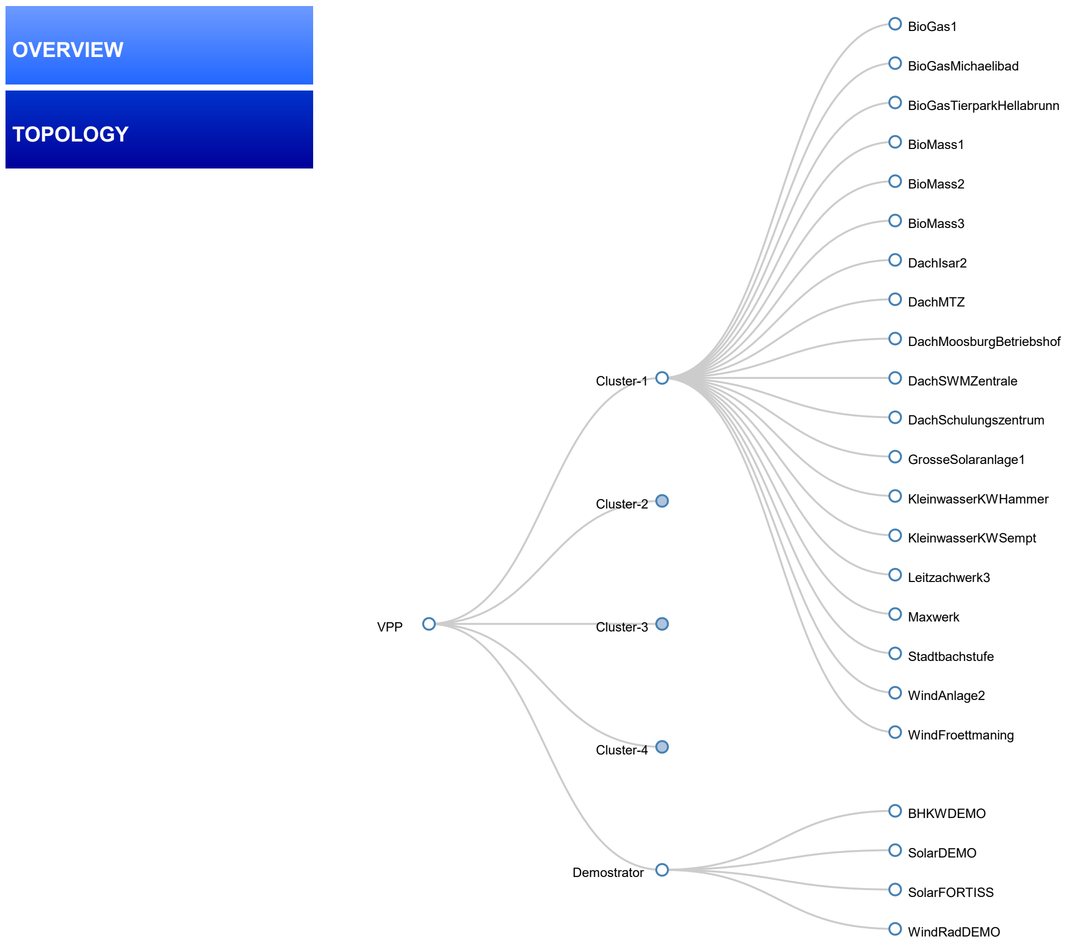


Figure 5.18: The topology of the co-simulation (only partly expanded due to space limitations).

Further refinements and further usage

The demonstrator was improved and refined after its first usage, where more features and components were added to expand the functionality. The general architecture remained unchanged. The new component setting includes storage components that add further complexity for the coordination algorithms, as time dependencies of the states are added. The demonstration table and the HMIs were adapted to show the increased complexity with storage (Figure 5.19).

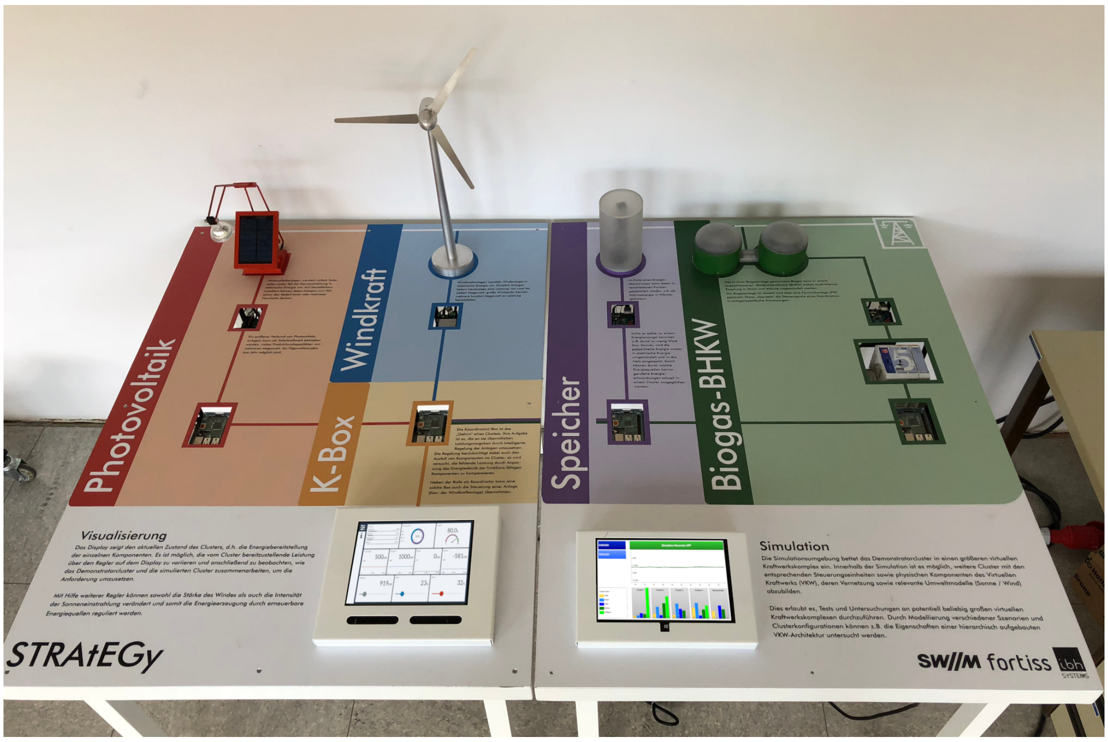


Figure 5.19: The second version of the demonstrator with storage components.

The training centre of the utility company SWM further uses the demonstration table, in particularly the architecture of the components and their interaction, including the embedded devices, like Arduinos and RaspberryPis. The table is also part of their future lab and has been shown according our partners on different innovation fairs. According their developers it is a good example to demonstrate the integration of components and the feasibility of hierarchic architectures. Hence, we show in this case study with our first demonstrator version and the refinement process that our hierarchic architecture is feasible and can be applied by other system engineers.

5.5 Discussion

In this section we discuss and answer our case study research questions, by reflecting our approach and the technical implementation. We further present the limitations of our approach and discuss the threats to its validity. We also include the received feedback from our industrial partners.

5.5.1 Research Questions

RQ1: How does the hierarchical concept affect the designs of EMS and the architecture of composed energy coordination systems?

The integration of components (devices or more complex EMS) relies on the available interfaces that describe the input and output of the components. Interfaces are the only way in which a software system interacts with its environment. They are therefore the central aspect to define an architecture. The behaviour of the components, thus, the understanding how the input is used to generate the output, is important to describe the functionality of the system. This is the second important specification activity of describing an architecture. The third part is the composition of the systems that describes how different components are interconnected. It clarifies which outputs are connected to which inputs and how this affects the overall system behaviour.

All three activities are described in our case study in the previous sections. We were able to provide the necessary specifications using the atomic behaviour cases that we introduced in Chapter 4. They guide our specification activities quite successfully starting from simple monitoring use cases over to a coordination model and further to an aggregation and delegation approach. The atomic cases supported us to create a meaningful abstraction for the EMS interfaces. Our approach also demonstrated the possibility to refine the interfaces, first by stating what a flexibility is, and then extending the flexibility to a time-series graph, which can be aggregated more accurately as the initial version.

In summary, the hierarchical concept based on the composite pattern leads the necessity to define an interface of an EMS that is used at multiple hierarchical levels. The atomic cases provide a good orientation to start the specification and further guidance leading to the description of a hierarchic architecture including the three important specification goals: component interfaces, component behaviours and composition.

RQ2: Does the hierarchy support different energy coordination mechanisms?

Section 5.4.2 introduces the coordination behaviour with the previously introduced interfaces. The goal of the coordination is to reduce deviations from the schedule based on flexibilities. We present three coordination algorithms. Each algorithm results in slightly different coordination signals and different timings in activating the available flexibilities. Therefore, the operation varies in some parameters, such as the speed to reach the coordination set-point and related costs. Nevertheless, the question if the hierarchy supports different coordination mechanisms based on the same interfaces can clearly be answered with yes.

Further, when considering the different coordination algorithms, starting from very simple activation rules towards a more sophisticated mathematical model with an MPC, we observed promising refinements for the interfaces. Particularly, the MPC helped us in refining the specifications of the flexibilities. Therefore, we see that mathematical models are beneficial for the definition of the interfaces and should be part of the specification activities. Current activities for defining the interfaces often rely purely on the device parameters (e.g. the VHPready standard (VHPready, 2017)). In consequence, more abstract information models, like the one presented for the flexibility, are not provided. The device parameters are sufficient, as long as the integration focus on devices. However, with the transition to more general EMS, as currently discussed in research and by the industry, we need to combine mathematical models with the current technology. This is further elaborated in the next chapter.

Finally, in this case study we discuss a top-down coordination, meaning that the set-points are calculated by a higher level system. But our approach is not limited to a top-down coordination. It is easily possible to realize distributed coordination with the same approach, where each component does not offer its own flexibility but expects a system price to decide its own set-points. We can use the same atomic steps to define the related interfaces, behaviour and composition creating a hierarchic architecture. But we need to understand two behaviours, one that describes how a component will react on a price and another one that describes how the prices are calculated and which information is required for that (e.g. forecasts, price sensitivities, etc.).

RQ3: How do the proposed architecture design patterns help us with the development of such coordination systems?

The last research question targets the validation of our architecture design patterns and the technical feasibility. In Section 5.4.5 we demonstrated that we were able to implement the hierarchic systems and that the chosen architecture design patterns were helpful. The precondition for that was a clear specification of the interfaces together with the coordination behaviour based on the composite pattern as presented in the Sections 5.4.1 - 5.4.3. Further, the relation of the architecture design patterns to the implementation decisions are shown in Section 5.4.4 in Figure 5.13. Therefore, the necessary implementation decisions are in line with the architecture design patterns. Another finding was that the *client-server* model requires a decision whether the EMS interfaces upwards are realized as a server and the higher level ClusterHead reads those values with a client, or if the client-server model is turned around. Both approaches are possible. We recommend to have an EMS server with an interface upwards the hierarchy as presented in Section 5.4.5. The reason is that our architecture should support the possibility for monitoring so that we receive an assurance that an EMS behaves as expected.

Further, we were able to implement the hierarchic coordination with available technology, meaning that it was not necessary to develop new technical protocols. Available approaches, such as REST (*unified interface*) and the plethora of available libraries for servers and clients are sufficient and quite successful in implementing such a system (see Section 5.4.5). Therefore, the focus on designing coordination systems should not be on the development of new protocols, but rather on a clear specification of the data at the interfaces. The constraints from the *composite pattern* were helpful to specify a hierarchic architecture for the coordination of EMS (as shown in Figure 5.2). The *layered systems* architecture design pattern is helpful to ensure that the EMS does not expose all internal data. It allows also to consider state-of-the-art approaches to add security. Finally, the *publish-subscribe* pattern was not used in this case study. We refer to a following case study to understand how the features of this additional pattern help to improve the system further.

Further findings

To create the case study we used the SESSIM co-simulation framework. It was very beneficial not only to create the final demonstrator, but actually is was used all the time, particularly to develop and test the interfaces and coordination algorithms at much earlier stages. The advantage of using this framework was that we were able to create a simulation of the expected scenario very early. We could also start with much simpler interfaces and refine them up to a stage, where we were sure that the coordination will work with real components. The framework further offers the advantage that it enforces us to develop a communication that is based on inputs and outputs of actors, which are not sequential and share no common data states. This is very close to real systems. SESSIM even allows that each actor has its own server with a dedicated interface. This saves time, because testing can be done early and without the necessity of deployment the code on distributed infrastructure.

5.5.2 Threats to Validity

In this section we discuss the main threats to the validity of our case study results.

Construct validity: Construct validity concerns the assurance that we correctly operationalised the phenomena in which we are interested. To answer our research questions, we followed a structured approach to define a hierarchic architecture for the composition of energy systems based on the assumption that an EMS is the major component to interact with. While the approach seems reasonable some threats remain. Current VPPs compose energy systems in a central architecture, not a hierarchic one. The reason is that VPPs are software systems operated from a central place. Those software systems run on hardware with good computational power and their performance limitation depends on the number of connected components and algorithms that are used to provide control within a VPP. Currently, VPPs are used more for data collection and monitoring issues. The pressure for control is still not that high. However, it is clear that the pressure for control increases in the foreseeable future. The question is how big the pressure becomes and how intense the control will be so that central optimization becomes impractical. Only if the central control becomes impractical, hierarchic groups might be a solution, but decentralised control could be another solution as well. Therefore, the biggest threat is the question, if hierarchies are required at all. We cannot answer this question completely at this point. However, if hierarchies are considered as a solution, our case study shows how the design of EMS interfaces can be approached and how this enables coordination of energy systems. Finally, since the specification of interfaces are an important engineering task for future energy systems, our approach helps to lift the specification from single device to more complex EMS with higher data abstractions at their interfaces.

Internal validity: Internal validity concerns the assurance that the case study delivers correct results, such as cause-effect relationships or the data extraction and analysis. This includes the validity of the data used, the validity whether the models calculate results that are realistic.

With the VPP investigated in this use case, we made several simplifications that might not contain all the details that real systems require. For instance, each component provides a maximum production capacity, but not its lower limit. This is used to calculate flexibilities. Hence, the flexibilities would be different when the lower limit is considered as well. Further, we have also not defined further details that are required for coordination. For instance we assumed that the schedule is given. But we neglected how they are created and how users can influence them, for instance how they activate maintenance modes. Therefore, it is obvious that our case study is limited, since the system interfaces need to be extended for real applications. But this is a matter of interface details and we have shown that our interface can be refined. We think that the VHPready specifications (VHPready, 2017) already go in the right direction, since they cover most of our points and provide many more. However, they are not able to yet to describe flexibilities. Hence, they need also to define more abstract data types and our approach would be beneficial for that specification. Particularly this abstraction is also necessary to lift the interfaces from single devices to groups of devices and more complex EMS.

Further, our case study works with generalized data provided by the utility partner. It means that we have normalized historic production profiles of the component's types, but no individual data that can be used to reason about schedules. Our flexibilities are only examples to provide the possibility for control. We had no previous data to analyse them in more detail. Hence, they are not used to provide quantitatively reliable systemic statements, but to demonstrate how interfaces play together with coordination approaches and hierarchic structures and the technical feasibility. To derive a coordination scenario we assume instantaneous deviations from the schedule to show the reaction of our coordination algorithm. We assume a one second delay that covers the control decision and communication. This not necessarily exact, since delays exist due to the TCP/IP communication and the internal processing time. However, this assumption seems reasonable to explain the three different coordination approaches and their effect.

External validity: External validity is concerned with the generalizability of the case study. Our system investigates energy system coordination of EMS in the setting of virtual power plants. Three algorithms with increasing computational complexity are compared. However, there are many more coordination approaches available. They rely on their own models and require specific data to either determine the set-points (in a top-down coordination) or a guidance signal such as prices (in a bottom-up coordination). Our interface and coordination is currently limited to the case study functionality. To make any claims about the applicability of the other coordination approaches, we need to repeat the specification process. However, we are quite confident that we are able to do that for centralized and decentralized approaches. To investigate other scenarios and improve the generalizability of our work we have carried out a second case study with a slightly different coordination focus in Chapter 6.

5.5.3 Limitations

In this section we discuss the limitations of our approach and our case study results. Some of these limitations can be obviated by extending our specifications or improving coordination algorithms. Other limitations are more fundamental due to the assumed hierarchical structure and the specific context of the system under analysis that hampers to transfer our approach to other types of systems.

Quantified results: Our case study uses real historic data from the industrial partner to demonstrate the operation of a VPP. We also use real data to parametrize the size of components within the VPP. However, to demonstrate the coordination we used exemplary data for the flexibilities, such as ramps and costs. The reason is that so far flexibilities are often described in research, but there is no reference for a flexibility and no data is available for that. Also for the MPC, we used exemplary replacement costs to calculate the set-points. Therefore, we cannot make any statements about the quantified savings when such a coordination is used, since we focus more on the technical questions to create such a system[6].

Applicability in practise: During a last workshop with our industrial partners from the Munich utility we discussed the possibility to combine our hierarchical approach with the current solution. Two feedback remarks were particularly of interest as they show the current practical limitations. First, the clustering of components becomes of interest, if the current central solution encounters performance limitations, for instance due to long optimization time. This was not yet the case at the utility, since their VPP had 120 components and most of them were not coordinated due to regulations that prevent the usage of flexibilities (e.g. solar, wind and water without hydropeaking due to ecological reasons for fishes). So comparably, only a small number of components are actively controlled, most of them provide only data to improve forecasts and the trading capabilities. But the utility carefully follows the topic, particularly because the need for control is growing with more renewable energy sources and the effect of electric vehicles. Second, current devices are integrated directly, for instance with a FW or protocols that are provided by the component providers, such as the VHPready standard. Distinct EMS exist for some use cases (like PV+battery combination), but they focus more on their internal optimization and often lack interfaces to interact with them. However, interactive EMS are being developed now in research and open initiatives like OpenEMS[7], so the question of integrating them is not a question if they will be integrated, but when and in which context, for control or economic reasons.

[6]Note, to obtain better quantifications we refer to the second case study in Chapter 6

[7]`https://openems.io/`, last accessed March 2020

5.6 Summary

This chapter investigates the hierarchical architecture for EMS coordination in the context of VPPs. We explain the concept of a VPP and that today's solutions use a central approach, with its advantages and disadvantages. To increase the flexibility we describe how VPP components are extended into a hierarchic architecture using our atomic behaviour steps and the proposed architecture design patterns. Therefore, a component is represented as an individual EMS that hides the internal implementation details and provides only abstracted information for energy production and the flexibilities at its interface. The ClusterHead that is responsible for the coordination of the EMS. It collects the data and provides the aggregates to the next higher layer. Additionally, it sends control signals to keep the desired schedule. For that, we use only the available flexibilities and calculate control signals, which contain the set points for next times steps of the components. This allows us to control individual components and groups of components in the same manner. Our assumptions for this approach is that components comply with their flexibility specifications and follow the control signal. We describe the implementation of the presented concept as a demonstrator and explain the used architecture design patterns. The demonstrator contains embedded devices, the ClusterHead software, which is deployed on a RaspberryPi, and the connected SESSIM co-simulation framework, that additionally simulates further virtual components to show the interaction in a larger setting. The demonstrator table is further used by the industry for additional trainings and as an innovation example.

6 | Architectures for Coordination Systems of Quarters

In this chapter, we describe a second case study to reinforce the generalization argument of our approach. In this case study we investigate the architectures for coordination systems of quarters that integrate EMS with multiple energy networks. We also extend our activities to align our approach stronger with mathematical modelling in order to demonstrate how sound energy system models support the specification of architectures and EMS interfaces. This case study was carried out with several industrial and research partners.

We start this chapter with a short context to introduce and clarify related terms, particularly, multi energy systems and sector coupling. We also motivate the importance of this topic. We describe the current technological solutions that are used for this context, the challenges and the gap between the available technologies and the desired solutions. From this starting point we proceed with our case study goal definition and the formulation of our research questions. Upon that we describe our case study design and proceed with the case study execution, where we show stepwise how the EMS coordination is achieved from the mathematical point of view, how we define the architecture, the EMS interfaces and behaviour based on those models and finally the implementation into a Multi Energy Management Aggregation Platform (MEMAP). After the implementation, we carry out a quantitative comparison with and without our coordination solution. After that we evaluate the numerical results and provide a discussion. A short summary with our contributions is presented at the end of this chapter.

6.1 Context of the Study

The power sector in Germany has tremendously increased its share with renewable energy in the recent years. Today, around 54 % are produced from renewable resources[1]. It is foreseeable, that the share will further increase in future. This environmental sustainable development is currently, however, mostly restricted to the electricity sector. Other energy sectors, including heating and mobility, still rely on fossil fuels. Consequently, sector coupling activities try to improve this situation by combining components of multiple energy networks. The German Government describes in their Climate Protection Plan 2050 (BMU, 2016) sector coupling as one of the major activity areas for future development. Sector coupling integrates the different energy areas and facilitates the usage of (renewable) energy sources for heating, cooling or mobility instead of fossil fuels, which improves the overall reduction of green house emissions. Sector coupling provides also an additional degree for flexibility, which allows a better utilization of energy resources.

Acatech and Leopoldina (Acatech, 2017) particularly emphasise the importance of sector coupling if buildings are involved, because they strongly benefit from better heat production and technologies, like heat pumps, thermal storage, district heating, especially, when an integrated solution exists. Further, they emphasize that sector coupling helps also to utilize waste heat from the industry, which can be injected into local district heat networks, if such possibilities exist. This additionally improves the overall energy efficiency.

[1]Fraunhofer Energy Charts: `https://www.energy-charts.de`, accessed in March 2020

The combination of the technologies is an integration challenge. It involves the interconnection of the different devices for control, but also the interconnection of the different stakeholders of the systems, such as users, building owners and the operators of the infrastructures (e.g. of building automation systems, heating or electricity networks). The interconnection is also an economical challenge, since the costs and the revenues for the usage of the resources need to be settled as well. This balancing act between the technical and economical integration is challenging for system manufacturers, particularly, because their systems are originally designed to operate in a closed environment, but also because requirements for additional interaction are not clearly specified. We therefore use our approach for hierarchic coordination architectures to reach a clearer specification for the desired system and to build a prototype that is able to integrate such distributed technologies.

State of the Art

The development of sector coupling concerns mixed areas with residential, industrial and office buildings. Today, buildings in such an area have no systems in place to provide energy services to networks, at best, they have individual automation systems that provides a power management for single buildings. Providers of building automation system have a long history and offer a large variety of components, different controllers and software. Their automation system architecture (see Figure 6.1) is designed as SCADA including the different layers for field devices, automation with PLC components and the upper layer for the management software.

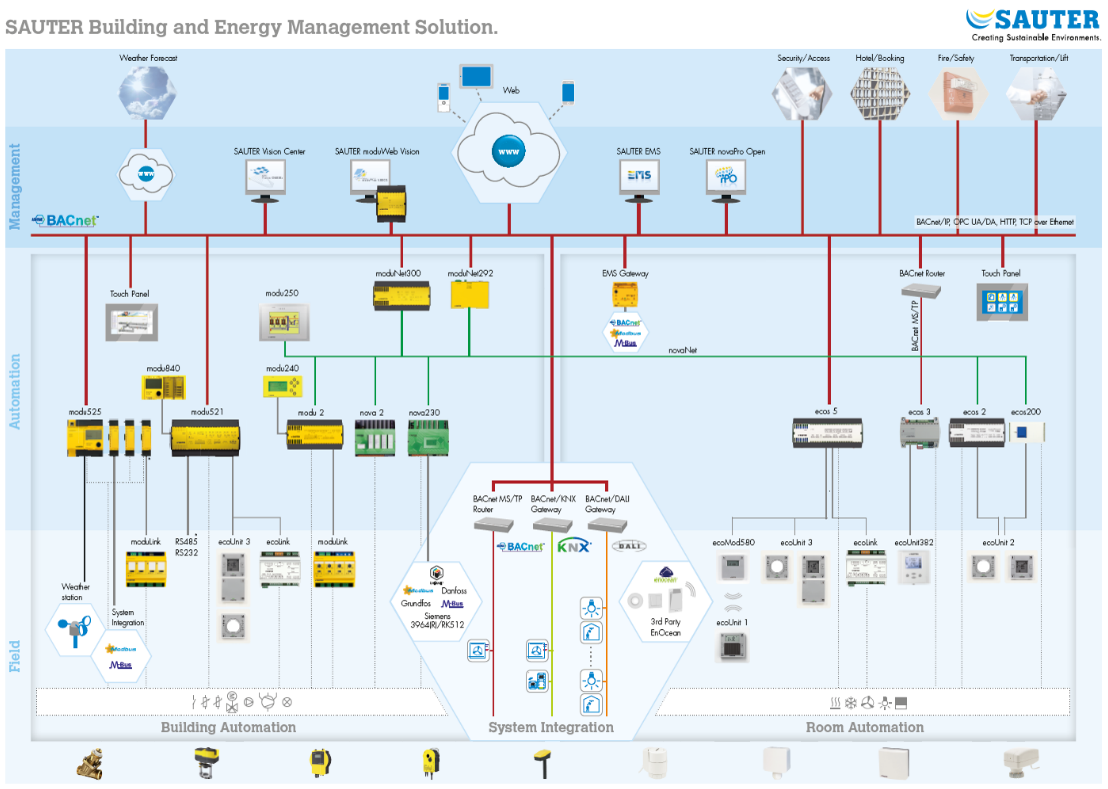


Figure 6.1: A commercial building automation system architecture (Sauter Cumulus GmbH, 2018).

The field layer encapsulates different sensors and actuators within the building, which are either directly connected to controllers, or communicate their values over additional modules. A variety of different protocols exist for the communication, such as BACnet, DALI, EnOcean, KNX, Modbus, Profibus, Zigbee, Z-Wave and many more. Next, we have the automation layer. It encapsulates the controller components, such as PLC or other programmable devices. They include the behaviour logic for control and forward important signals to the upper layer. The communication upwards is already realized over Ethernet with IP based protocols, such as TCP, HTTP(S), or OPC UA. The next layer is responsible for the management and configuration. It consists of different software components for visualization, statistical data processing, configuration or alarming. Depending on the vendor of the system, the management layer also offers software for remote access over VPN, WebServices (e.g. weather forecasts) and additional interfaces for external systems, for instance for security companies, booking services, or for the building's technical maintenance.

During our initial workshops with the vendors it became evident that the internal control logic as well as the internal parametrization is not only complex but also quite individual for every building. The reason is that the available sensors and actuators influence the behaviour of the equipment. For example the temperature control of a building strongly depends on them. Sometimes, there is an integrated solution, which receives the temperature values from integrated sensors inside the rooms and controls the heating or air conditioning correspondingly. This is a desirable situation. However, due to cost savings, building owners often use individual systems that are not integrated into the vendors solution, e.g. local thermostats, smart thermostats like NEST[2] or TADO[3]. Such solutions interfere with the solutions of the vendor and requires an adaptation of the control logic of the building automation. This is a complex situation, where an individual configuration is required, depending on available internal technologies.

Further, not only the internal control logic is individual for each building, each building has also an individual set of energy supply devices. Those devices are planned and installed to consider all kinds of conditions, even harsh ones at cold times in the winter. Our industrial partners confirm that the current planning is such that building's energy supply must be fully available also at -14/-16°C. Consequently, the installed equipment has a larger capacity as required during the most time of the year. Therefore, it often operates in a non-optimal range and its full efficiency is seldom used. To reach a better utilization, it makes sense to combine several buildings in a quarter. This allows to use the installed equipment more efficiently and also better utilize the modernization of equipment in such a combined quarter. With integration, modern equipment is used more frequently and pays-off more quickly leading to a faster substitution of less efficient, potentially less environmental friendly equipment.

Building technologies have multiple energy networks. They combine electricity, heating, cooling, (hot) water and in future also electric mobility. The different energy networks have components that act as producer, consumer, storage to a particular energy network. There are also components that act on several networks at the same time, such as CHP or heat pumps. Hence, each building can be understand as a multi energy prosumer system. To integrate such buildings, we need a multi energy coordination system as envisioned in our case study.

[2]`https://nest.com`, accessed in October, 2019

[3]`https://www.tado.com/de/`, accessed in October, 2019

6.2 Study Goal and Research Questions

In this section, we present the goal of our case study and derive again several research questions to guide our work. In Chapter 1 we stated three research objectives. The case study in Chapter 5 handles the first objective. This case study focuses on the second and third research objectives.

6.2.1 Study Goal

The research objectives two and three from Chapter 1 are as follows. The former is to **develop a coordination approach for EMS based on mathematical models using the specification elements of objective one**; and the latter is to **identify the benefits but also the drawbacks and limitations of hierarchic energy coordination systems.** We use those objectives to state a structured study goal using the goal definition template as proposed in Wohlin et al. (2012):

We analyse	*the transfer of mathematical models into a coordination approach*
for the purpose of	*understanding the process and demonstrating the necessary steps*
with respect to	*specification of system boundaries, interfaces, architectures*
from the viewpoint of	*software developer and system integrator*
in the context of	*multi energy districts.*

The identification of the benefits, drawbacks and limitations is a reflection of the case study work to understand which points are helpful at which system development stages and which are not.

6.2.2 Research Questions

Based on our study goal we derive three research questions that we answer throughout the following work.

RQ4: When energy models are available, what are important details from those models to specify system interfaces and what are the possible transformation steps to develop an architecture of a coordination system?

In energy systems control and optimization applications rely on mathematical models of power system engineers. Simultaneously, software engineers that create control and monitoring systems are faced with questions for designing appropriate system interfaces, such that different kinds of interaction schemes are supported. Often, the required data for the interfaces are implicitly available within the mathematical models, or at least, mathematical models have certain expectations about the desired data to enable an optimised operation. Moreover, control and optimization applications require data to be transferred to be able to operate. But this knowledge is not always transferred and both tasks remain two separated processes. Therefore, mathematical models for optimization often remain on paper (or as MATLAB code). With our research question we try to find out the possible steps for that transformation. The first part of the question addresses the transfer ot the model into a specification of system interfaces. This requires to find the necessary data points for communication due to that model and to clarify whether there are different representations of a model that can be used for that purpose. Further, this research question also requires to clarify the meaning of the term system boundary, to understand whether there is a different understanding of this term with respect to power system models and coordination systems and whether this leads to problems. Once the first part is clearly understood, the second part of this research question aims to investigate which architecture design decisions are necessary to create a system that is based on the chosen coordination model with the chosen interface data. It also considers the specification elements of the previous case study, including the hierarchical structure, the chosen architecture design patterns and the atomic behaviour cases that help to describe the interaction of the coordination system with other EMS.

RQ5: Which systemic effects can be expected from coordination systems?

In our first case study, we do not provide quantified results that explain how coordination and hierarchic architectures improve the current power system operation. This is made up here. We describe a concrete scenario of a district that operates with and without a coordination system. We compare the results for both cases to evaluate the potential of combining multiple EMS, in order to improve the state of the art by utilizing the available over capacities from individual buildings.

RQ6: What do hierarchic coordination systems achieve in the context of SES and EMS and what do they not achieve?

The question reflects our general motivation to clarify whether our approach delivers what was promised in the beginning or not. The initial idea was to investigate new energy system architectures that provide control not with a top-down approach (as done today with few large power generators), but to provide control with a bottom-up approach to reduce complexity. The bottom-up approach envisions to integrate many distributed systems that offer similar options (supply, demand, storage) in a hierarchical architecture to support the handling of individual components and group of components in the same way. With this question we intend to answer whether this is feasible. We further aim to understand what is improved with the hierarchical approach and what is not. For instance we would like to know, if our approach supports only centralised control decisions, or whether decentralised control is supported as well.

6.3 Study Design

To answer the case study research questions we again choose an experimental approach, where we provide a technical prototype that interconnects several EMS. The prototype is developed as part of the Multi Energy Management and Aggregation Platform (MEMAP) project. The aim of the platform is to interconnect several building on the local level, in order to enable a potential energy exchange. MEMAP develops the required interfaces and a software system for such an information exchange using a hierarchical architecture. It collects the data over consistent EMS interfaces (composite pattern) and uses them for the coordination. The interface provides only abstracted information of the building's capabilities, not all the details, since only information for the building's behaviour from the energy point of view is necessary. The building automation system still controls the internal components itself and considers its local constrains. The only task of the platform is the creation of an optimized schedule, so that connected buildings save costs / emissions, if more efficient components for energy production are available.

Before we start with the schedule generation and the platform's interface specification, we introduce the general system architecture for a better understanding about MEMAP's operation. MEMAP is developed to operate as a system that runs on an external server and collects data from the buildings. Its task is to improve the energy efficiency in the neighbourhood. Therefore, it establishes a connection to a number of buildings, or more precisely, to the building's EMS interfaces that provide the required information (Figure 6.2). For external systems MEMAP is also seen as an EMS. Hence, it provides also an interface with the same specification as its children. The data from the interfaces provide the available energy resources of the buildings. They represents an abstract view on the components that provide potential power injections for the different networks types, e.g. heating and electricity. The view reflects the current system state to enable monitoring and provide options for the near future for planing. This allows MEMAP to monitor and plan an optimized operation of the connected buildings.

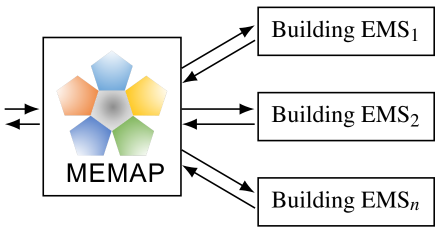


Figure 6.2: MEMAP's high level system architecture.

MEMAP's high level architecture is similar to the conventional SCADA pyramid structure with the management level at the top layer and the field devices at the bottom. However, there is a crucial conceptual difference between MEMAP and SCADA concerning the expected control logic. SCADA envisions that the logic for control, planning and operation is established on its management layer. Components on the lower automation and field layers are only responsible to execute commands and to keep threshold values. Therefore, in SCADA the upper layers are more "intelligent" than the lower ones. In contrast, MEMAP is not designed to control a large variety of different field components of a system. Instead it operates on a unified and abstracted specification of the buildings' resources. They provide the data to MEMAP, which collects them, aggregates them and calculates an optimized schedule for the operation to increase the combined energy efficiency of the buildings. The new schedule is communicated to the buildings as desired set-points. The execution of the schedule is left to the EMS. The intention is that following the schedule optimizes the efficiency of overall production and consumption, because the more efficient components are preferable activated. Hence, the intended cost reduction is a good incentive for collaboration.

In consequence, most of the control logic is left to the building's EMS, which are much better suited to monitor and control its internal equipment. This has several positive effects. Each building EMS remains a completely individual system, with its own equipment thresholds, control logic, user access rules, firewalls and other configuration details. The connection between MEMAP and the EMS is very loosely. Connected systems can be stopped, restarted, updated and redeployed independently. This supports redundancy and increases the robustness. Such a loosely coupling is in fact necessary, to allow independent system development and with that the usage of MEMAP for multiple EMS and different vendors of such systems.

Further, current building automation systems rely on a set of common communication protocols (see also Section 6.1). Our case study is designed to use the OPC UA[4] protocol. The OPC UA communication is a widely used protocol from the domain of industrial automation. Control systems and field devices of building automation systems support this protocol quite often. Therefore, the usage of the protocol towards higher automation systems, as intended by MEMAP, is part of the study. The benefit of that protocol is that it brings much build-in functionality such as security with authentication and authorisation, client-server instances, publish-subscribe mechanisms, look-up mechanisms to access data, look-up mechanisms to access the semantics of data, method execution, etc.

6.4 Study Execution

This case study combines mathematical models, hierarchic system architectures and industrial protocols in a demonstrator. We start the case study execution by defining the mathematical model for coordination. This model serves us as a preparation to derive the interfaces from that model and to define the system behaviour based on that model, which we specify in our second step. The third step shows how such a component is able to interact in a hierarchical system by defining the aggregation and delegation behaviour. Further, in a fourth step we demonstrate our implementation, showing our co-simulation approach and its extension towards a platform with the OPC UA interfaces. In our fifth step, we finally show an execution scenario, where we show an exemplary district with multiple multi-energy prosumer buildings that exchange energy. This scenario yields quantified results to demonstrate a potential of such coupling.

6.4.1 Step 1: Definition of the Coordination Model

Energy system coordination relies on the usage of mathematical models. We introduced several well known examples in Chapter 3, where we described how those models are created and which purpose they have. For this case study we have developed a classical approach that is similar as the short term planning approach (see equations 3.4) and relies on the transfer of energy. The development of our model was a combined work of several researchers[5] that was presented in Bytschkow et al. (2019). We use a model that goes into the same direction as the energy hub approach developed by Geidl and Andersson (2007) by using multiple power injection vectors from multiple energy networks. The general form of the model is given by the following objective

[4]OPC Unified Architecture (OPC UA) standard (OPCFoundation, 2017), which is the successor of the Open Platform Communication (OPC) interface specification. OPC UA replaces all previous OPC based interface specifications like OPC DA (Data Access), OPC HDA (Historic Data Access), OPC A/E (Alarms and Events), etc. and makes the interface platform and DCOM independent.

[5]We particularly thank Alexandre Capone and Jan Mayer amongst others for developing and testing the model together within the MEMAP research project.

function with the related equality and inequality equations:

$$\min_{\mathbf{x}} \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{x} \tag{6.1a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{6.1b}$$

$$\mathbf{G}\mathbf{x} \leq \mathbf{h} \tag{6.1c}$$

$$\mathbf{x_{lb}} \leq \mathbf{x} \leq \mathbf{x_{ub}}, \tag{6.1d}$$

where we have the vectors $\mathbf{b}$, $\mathbf{x}$, $\mathbf{h}$, $\mathbf{x_{lb}}$, $\mathbf{x_{ub}}$ and $\boldsymbol{\lambda}$ represent the required power demand, the power generation, the constraints from storage, the lower and upper boundaries of generation and the cost vector, respectively. The matrix $\mathbf{A}$ describes to which extend each generation unit contributes to the demand and and $\mathbf{G}$ how temporal effects are considered during the different time steps. By applying the optimization model repeatedly for each time stamp with a time horizon of length N_H, we obtain a classical MPC as described by Rawlings and Mayne (2009).

For the coordination of buildings, the power network is described as a set of linear equations and discretized by sampling the time with time steps of length Δt. We assume that we have a quasi static system for electricity and heat exchange, meaning that dynamics caused by effects such as mass flows are negligible for our coordination time frames. We further assume that our network is lossless, meaning that power is exchanged and used only by the system nodes (the buildings), not by the network that transports that power. With that our power is conserved within the network for each time step t:

$$\sum_{i=1}^{n} P_{\xi,i}(t) = 0 \tag{6.2}$$

where n represents the number of buildings and $P_{\xi,i}$ the individual power injection of building i and $\xi \in \{Q,P\}$ represents the heat network (Q) and the electrical network (P). The individual building's power injections are calculated as:

$$P_{\xi,i}(t) = \sum_{j=1}^{n_i} \eta_{\xi,j} G_j(t) - D_{\xi,i}(t) + S_{\xi,i}, \tag{6.3}$$

where n_i represents the number of generation devices of building i, G_j the generation unit j within that building, $\eta_{\xi,j}$ its efficiency rate, $D_{\xi,i}$ the demand and $S_{\xi,i}$ the power injection of the storage devices within that building. Further, we consider that each of the power devices and storage is limited by the specified production constraints, i.e.,

$$0 \;\leq\; \eta_{\xi,j} G_j(t) \;\leq\; P_{\xi,j}^{max}(t) \quad \wedge \quad -S_{\xi,i}^{dis} \;\leq\; S_{\xi,i}(t) \;\leq\; S_{\xi,i}^{ch} \tag{6.4}$$

where $P_{\xi,j}^{max}(t)$ is the production limit for production devices and $S_{\xi,i}^{ch}$, $S_{\xi,i}^{dis}$ limit the charging and discharging. Note, that $P_{\xi,j}^{max}(t)$ is fixed for devices that operate on fossil fuels and variable for renewable sources that depend on the environmental conditions. This applies for both, the heat and electrical power. To consider the storage capacities, we further need to consider a time dependent charging process, i.e.,

$$-C_{\xi,i}(t_0) \;\leq\; \Delta t \sum_{t=t_0}^{N_H} \cdot \eta_{\xi,i} S_{\xi,i}(t) \;\leq\; (C_{\xi,i}^{max} - C_{\xi,i}(t_0)), \tag{6.5}$$

where $C_{\xi,i}(t_0)$ is the state of charge at time t_0 and $C_{\xi,i}^{max}$ the maximum available storage capacity. With those constraints, we can create the coordination with an MPC. For that we write the generation and storage into the **x**-vector as:

$$\mathbf{x}^T = (G_1(t_0),...,G_1(t_{N_H}),G_2(t_0),...,G_2(t_{N_H}),...,S_{Q,1}(t_0),...,S_{Q,1}(t_{N_H}),...), \tag{6.6}$$

where G represents the generation unit, S the storage units with the time steps $t_0,...,t_{N_H}$ that are considered for the time horizon of planning. Similarly as the generation, we write the demand, which shall be fulfilled, as vector **b**:

$$\mathbf{b}^T = (D_Q(t_0),...,D_Q(t_{end}),D_P(t_0),...,D_P(t_{N_H})), \tag{6.7}$$

where D_Q, D_P represents the heat and electrical demand, respectively. Note, the set $\{Q,P\}$ indexing the demands represents the available networks. It can be easily extended to cover even more than those two sectors, leading to an energy system model that might consider also cold networks or heat networks with other temperature settings. In that case, the demand vector will be extended as well. Further, the constraints of component's operational limits are described by $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$.

The cost function C (the objective function) captures the incidental costs:

$$C = \int_{t_0}^{t_{N_H}} \left(\sum_{j=1}^{n_j} c_j^{fuel}(t) G_j(t) \right) dt, \tag{6.8}$$

where the integral describes a time horizon $[t_0..t_{N_H}]$, n_k the number of available power generation sources, $c_j^{fuel}(t)$ the fuel costs that are time depended in case of varying costs or emission profiles. The expression is discretized to:

$$C = \sum_{k=0}^{n_H} \sum_{j=1}^{n_j} \left(c_j^{fuel}(t_k) G_j(t_k) \Delta t \right), \tag{6.9}$$

such that the cost vector $\boldsymbol{\lambda}$ is written as:

$$\boldsymbol{\lambda}^T = (c_1^{fuel}(t_0),...,c_1^{fuel}(t_{N_H}),c_2^{fuel}(t_0),...,c_2^{fuel}(t_{N_H}),...). \tag{6.10}$$

Further, to clearly differentiate between the demand and generation, we use the convention that all generation values are positive and all demand values are negative. With that convention we create our matrices. The coupling matrix **A** is composed of several sub-matrices based on available components:

$$\mathbf{A} = \left(\mathbf{A}_1, A_2, ... \right). \tag{6.11}$$

The general composition of each sub-matrix has been chosen to reflect the sector coupling, namely the heat and electrical power network. Therefore, we split the matrix in sub-matrices representing each network.

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{A}_{i,Q} \\ \mathbf{A}_{i,P} \end{pmatrix}. \tag{6.12}$$

To create the sub-matrices a number of different generation devices are available.

First, we have conventional fuel driven generators, such as oil heaters, gas heaters or diesel engines, which cover one particular network. They can flexibly adapt their generation. We therefore call them **controllable resources**. They have a certain generation capacity and an efficiency factor.

Second, we have volatile components, for instance wind, photovoltaic or solar thermic components. They generate power that depends on their installed capacity and the environmental conditions, such as wind speed or solar radiation. We call them **volatile resources**.

Controllable and volatile components lead to a diagonal form of the sub-matrices. For a horizon of N_H time steps we have:

$$\mathbf{A}_{Controllable} = \begin{pmatrix} -\eta & 0 & \cdots & 0 \\ 0 & -\eta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\eta \end{pmatrix} \in \mathbb{R}^{N_H \times N_H}, \mathbf{A}_{Volatile} = \begin{pmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix} \in \mathbb{R}^{N_H \times N_H}. \quad (6.13)$$

The difference in the MPC is that controllable components have constant constraints $\mathbf{x}_{lb}$ and $\mathbf{x}_{ub}$, while the constraints of the volatile components are profiles derived from forecasts.

Further, we have storage components that represent the third category **storage**. They can feed in and feed out power from the network. To keep positive **x** vector elements (for computation) we split the matrix in two parts. This indicates that charging and discharging can be seen as a separate process and **x** contains both contributions. Further, to have a correct model of the system, we need to make sure that storage components avoid charging and discharging at the same time. We achieve this with small operational costs of the storage. The matrix for storage is then:

$$\mathbf{A}_{Storage} = \begin{pmatrix} -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & -1 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{N_H \times 2 \cdot N_H}, \quad (6.14)$$

where the first part with a negative sign represents generation that covers demand. This means discharging for storage components. The second part with a positive sign adds the need for other generation. This means charging for storage components. Note, each storage component has also efficiencies for discharging η_{dis} and charging η_{ch}. They are used for the calculation of the SOC with the **G** matrix, which will be described below.

Next, the fourth category are the devices that have an impact on multiple networks. These are for instance CHPs or heat pumps. We call them **couplers**. Their sub-matrices are similar to generation matrices, but have the double amount of rows. The diagonal elements contain the efficiencies for each sector:

$$\mathbf{A}_{Coupler} = \begin{pmatrix} -\eta_Q & 0 & \cdots & 0 \\ 0 & -\eta_Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\eta_Q \\ \\ -\eta_P & 0 & \cdots & 0 \\ 0 & -\eta_P & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\eta_P \end{pmatrix} \in \mathbb{R}^{2 \cdot N_H \times N_H}. \tag{6.15}$$

η_Q and η_P represent the thermal and electrical efficiency. Note, for CHPs, the efficiencies are available from the component's manufacturer specification. For heat pumps, the efficiencies can be easily calculated from the available coefficient of performance (COP). For instance, a COP of 2.5 means for heat pumps that 1 kW of electricity generates 2.5 kW heat. The η-factors in this case are therefore $\eta_Q = 2.5$ and $\eta_P = -1$.

The last sub-matrix addresses the delivery and supply from external networks. Similar to storage system, power can be taken out or put into the networks. We therefore split those two different parts, exactly as for storage systems. The matrix of external networks is therefore that same as for storage. The cost for the market is part of $\boldsymbol{\lambda}$ as well and depends on the market conditions. We consider that buying form the market corresponds to a given price profile (for instance according EEX[6]). Selling is also allowed and leads to a negative value, reducing the costs further. Due to grid tariffs, taxes and additional fees, we require that buying is more expensive than selling.

The remaining part are the capacity limits, which are time dependent and need an additional constraint formulation with the **G** matrix and the $\mathbf{Gx} \leq \mathbf{h}$ equation. We use triangular sub-matrices for that purpose:

$$\mathbf{G_{dis}} = \begin{pmatrix} \frac{1}{\eta_{dis}} & 0 & \cdots & 0 & -\eta_{ch} & 0 & \cdots & 0 \\ \frac{1}{\eta_{dis}} & \frac{1}{\eta_{dis}} & & 0 & -\eta_{ch} & -\eta_{ch} & & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \ddots & \vdots \\ \frac{1}{\eta_{dis}} & \frac{1}{\eta_{dis}} & \cdots & \frac{1}{\eta_{dis}} & -\eta_{ch} & -\eta_{ch} & \cdots & -\eta_{ch} \end{pmatrix} \in \mathbb{R}^{N_H \times 2 \cdot N_H}, \tag{6.16}$$

$$\mathbf{G_{ch}} = \begin{pmatrix} -\frac{1}{\eta_{dis}} & 0 & \cdots & 0 & \eta_{ch} & 0 & \cdots & 0 \\ -\frac{1}{\eta_{dis}} & -\frac{1}{\eta_{dis}} & & 0 & \eta_{ch} & \eta_{ch} & & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \ddots & \vdots \\ -\frac{1}{\eta_{dis}} & -\frac{1}{\eta_{dis}} & \cdots & -\frac{1}{\eta_{dis}} & \eta_{ch} & \eta_{ch} & \cdots & \eta_{ch} \end{pmatrix} \in \mathbb{R}^{N_H \times 2 \cdot N_H}, \tag{6.17}$$

where $\mathbf{G_{dis}}$ represents the limitations for discharging and $\mathbf{G_{ch}}$ those for charging. Accordingly, we define the vector elements $\mathbf{h}_{dis}$, which is the current SOC and $\mathbf{h}_{ch}$, which represents the remaining charging capacities. The elements of matrix **G** are zero, everywhere where storage components are not present.

[6]European Energy Exchange in Leipzig.

6.4.2 Step 2: Specification of the Interface and Behaviour

The second step for MEMAP is the specification of the interface. For the specification we use several input artefacts. On the one hand we had the mathematical model. On the other hand we had different requirement specifications that were prepared by our industrial partners, particularly from the producers of EMS. Those requirement specifications included several documents. One document was a mind map that described a list of data points from the EMS to MEMAP and vice versa. Additional documents were visualization mock-ups to demonstrate the usage of those data points. All three inputs were different in terms of precision and details. For instance, we observed that production units from our industrial partners were all equally described, no matter whether they represent controllable units, variable units or couplers. Also many details that are required for the MPC were missing. Therefore, to combine the mathematical model with the industrial partners expectation about the data exchange was a crucial task to create the desired coordination system.

In order to use the MPC for coordination, the correct level of abstraction is required. We start with the description of the data exchange between MEMAP and an EMS of a building, by specifying the input and output of the MEMAP platform. Due to the MPC that we have in mind we require one generic interface specification, such that all interfaces will have the same specification. The data communicated over that interface will of course differ. This shows one more time that the composite pattern is useful to create a modular approach for the coordination of SES. Before we go into the specification details we explain some foregoing considerations that are related to sector coupling, to understand the intentions behind our specification.

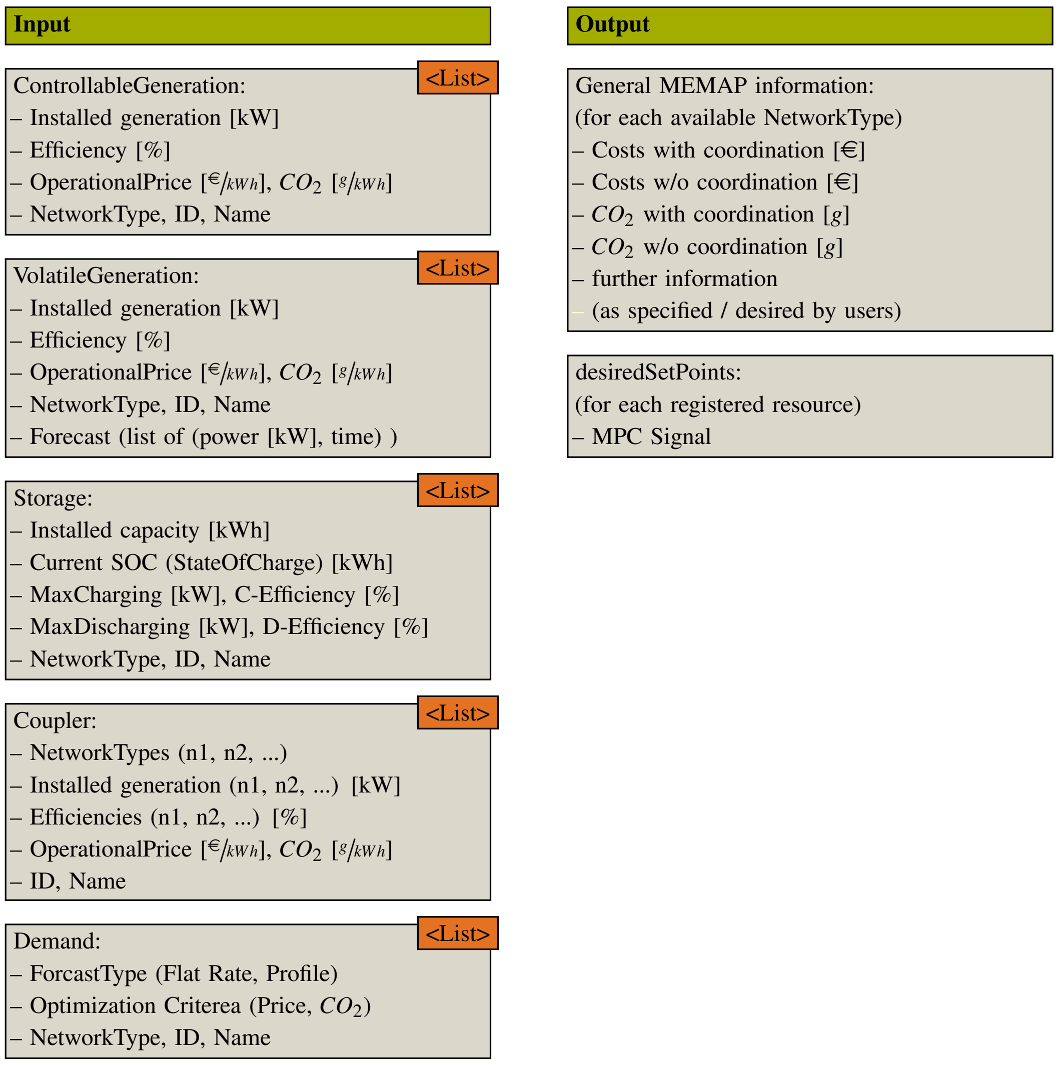
Preliminary considerations for multi energy systems and MEMAP

Any building or more complex quarter consumes or produces heat and electricity. It impacts therefore multiple networks. For the connection with MEMAP we use the classification type `NetworkType` to describe this. Electrical networks have a defined voltage level. District heating networks have a defined temperature level for the supply and return flow lines. For instance, the district heating in Munich is specified to 90°C for the supply and a desired value of 45°C for the return flow (SWM, 2015). To improve efficiency, district heating can be designed to operate with lower temperatures as well. The effects are described by Lund et al. (2014) and first installations are already in place (Wesche et al., 2017). This variety has to be considered by the MEMAP platform.

Further, each building provides components that can be classified according their type of operation, i.e. generation, consumption, storage or a conversion of electricity into heat and vice versa. In the following we therefore use the four types of components, which we also call resources: generators, here in particular also subdivided into `ControllableGeneration` and `VolatileGeneration`, `Storage` devices and so-called `Coupler`, which describe systems that have simultaneous effects on two networks, such as CHPs or heat pumps. The demand is defined as the fifth resource type `Demand`. Each resource shall have a unique identifier (`ID`), so that it can be referenced by the system. Finally, instead of addressing each component individually, the interface of MEMAP shall accept *lists* of typed resources.

Input and output specifications

In the next step, we specify the system interfaces of the prosumer systems on base of the input and output for MEMAP. This is the *Monitoring (L2-case)* and the follow up *Coordination (C4-case)* as presented with our case distinction for the specification of EMS. Since this case study is much more detailed as the previous one, we split the following specification in **planning data**, which is required for the MPC and the related coordination behaviour, and **live data**, which is required to display additional live information for the participants. The specification is summarized in Figure 6.3 for the **planning data** using the previously defined classes and Figure 6.5 for the **live data**. We explain it in more detail in the following.



Input

ControllableGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [€/kWh], CO_2 [g/kWh]
– NetworkType, ID, Name

VolatileGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [€/kWh], CO_2 [g/kWh]
– NetworkType, ID, Name
– Forecast (list of (power [kW], time))

Storage: <List>
– Installed capacity [kWh]
– Current SOC (StateOfCharge) [kWh]
– MaxCharging [kW], C-Efficiency [%]
– MaxDischarging [kW], D-Efficiency [%]
– NetworkType, ID, Name

Coupler: <List>
– NetworkTypes (n1, n2, ...)
– Installed generation (n1, n2, ...) [kW]
– Efficiencies (n1, n2, ...) [%]
– OperationalPrice [€/kWh], CO_2 [g/kWh]
– ID, Name

Demand: <List>
– ForcastType (Flat Rate, Profile)
– Optimization Criterea (Price, CO_2)
– NetworkType, ID, Name

Output

General MEMAP information:
(for each available NetworkType)
– Costs with coordination [€]
– Costs w/o coordination [€]
– CO_2 with coordination [g]
– CO_2 w/o coordination [g]
– further information
(as specified / desired by users)

desiredSetPoints:
(for each registered resource)
– MPC Signal

Figure 6.3: Input/output interface specification of MEMAP towards its children: planning data.

MEMAP expects as its input a number of different planning data points, so that it can create its optimization model and calculate its optimization result. The planning data consists of a list of resources, which are provided by the buildings for the coordination with MEMAP. We structure them in five different categories.

Controllable generation

The first category of planning covers `ControllableGeneration`. It represents conventional generation components that are particularly designed to supply power to one particular network. Typically, these generations units can be controlled within certain limits and are available over longer periods of time through the use of (fossil) fuels. Examples of such systems are oil and gas boilers, wood-fired heating systems and in certain cases, electricity generators such as diesel generators. We require several data points to consider such systems in MEMAP:

- Maximum power generation - in [kW]
- Minimum power generation - in [kW]
- Generation efficiency - in [%]
- Operational price - in [$^{€}/_{kWh}$] or additionally also CO_2 emissions - in [$^{g}/_{kWh}$]
- Network type - e.g. an ENUM of {electricity, heat (with temperature constraints), ...}
- ID and a readable name

A possible concrete representation is:

Name	Gas boiler
MaxPowerGeneration	20 kW
MinPowerGeneration	0 kW
Efficiency	95%
OperationalCost*	5.91 $^{ct}/_{kWh}$
CO_2 emission	0.20 $^{g}/_{kWh}$
NetworkType	Heat (≤ 80°C)
ID name	individual URI

*Note: the specification of an efficiency ≠ 100% affects the final costs, which consider the price and the efficiency. In this case the gas boiler would require 1.053 kWh of primary energy to produce 1 kWh of heat. This corresponds to an operational cost factor of 6.22 cent/kWh. The efficiency is an important factor for the economic performance of the component.

Following components that are used within buildings belong to the category of `ControllableGeneration`:

Name	Primary energy source	Production
Gas boiler	Natural gas, biogas	Heat
Oil boiler	Oil	Heat
Solid fuel boiler	Wood, pellets	Heat
Emergency generator	Gasoline, gas	Electricity

Volatile generation

The second category is `VolatileGeneration`. It describes the intermittent generation of components like photovoltaic, solar thermic components or wind generation. Volatile generation relies on the installed capacity and forecasts, that are dependent on environmental conditions such as solar radiation, temperature or wind power. They represent the upper boundary of the available production. Generation forecasts are generated from weather forecasts and the parameters of installed equipment. Historical information are helpful to generate forecasts as well, which we demonstrated in our research group in different projects (Rottondi et al., 2015; Bajpai, 2018). For the coordination system interface, we abstract volatile generation to a `ForecastedProfile`. The forecast profile is a time series of percentage values of the installed capacity. All other attributes are the same as for `ControllableGeneration`. The following attributes are necessary:

- Maximum power generation - in [kW]
- Minimum power generation - in [kW]
- Generation efficiency - in [%]
- Operational price - in [$^{€}/_{kWh}$], CO_2 emissions - in [$^{g}/_{kWh}$]
- Network type - e.g. an ENUM of {electricity, heat (with temperature constraints), ...}
- ID and a readable name
- Forecast profile - in [% over time]

A possible concrete representation is:

Name	Solar thermal system
MaxPowerGeneration	5 kW
MinPowerGeneration	0 kW
Efficiency*	95%
OperationalCost	0.0 $^{ct}/_{kWh}$
CO_2 emission	0.0 $^{g}/_{kWh}$
NetworkType	Heat (≤ 80°C)
ID	individual URI
Forecast	(0:00, 0%) , (6:00, 5%), (6:15, 7%), (6:30, 10%), (6:45, 5%), ...

*Note: the available power generation is limited by the forecast and the installed capacity. Further, the operational costs of volatile producers is usually zero, therefore, the efficiency factor of volatile producers does hardly impact the coordination.

Following components that are used within buildings belong to `VolatileGeneration`:

Name	Primary energy source	Production
Photovoltaics	Solar radiation	Electricity
Solar thermal system	Solar radiation	Heat
Wind turbine	Wind	Electricity
Small water turbine	Water	Electricity

Storage

The third category addresses `Storage`. Storage is available in form of electrical batteries and thermal energy storage, for instance hot water or phase change material devices. EMS with such options are able to store the energy for the appropriate `NetworkType` and release it at later time points. This category gives additional options for the scheduling by shifting the time of charging and discharging. The required attributes for scheduling are:

- Installed storage capacity - in [kWh]
- Current state of charge (SOC) - in [kWh]
- Maximum power for charging and discharging - in [kWh]
- Efficiencies for charging and discharging - in [%]
- Network type, ID and a readable name

A possible concrete representation is:

Name	Lithium-ion battery (one-phase connected)
Capacity	12.0 kWh
Current SOC	5.5 kWh
MaxCharging	3.3 kW
MaxDischarging	3.3 kW
EfficiencyCharging	98%
EfficiencyDischarging	98%
NetworkType	Electricity (220 V)
ID	individual URI

Following components that are used within buildings belong to `Storage`:

Name	NetworkType
Buffer vessel (various technologies)	Heat
Domestic hot water (various technologies)	Heat
Phase-change materials (various materials)	Heat
Batteries (various technologies)	Electricity

Coupler

Components that impact multiple networks at the same time are denoted as `Couplers`. Among these are, for example, CHPs, heat pumps and fuel cells. Cooling systems can also be classified as couplers. But their consideration is only beneficial if cooling production and demand can be measured and regulated and therefore modelled as an individual network type. Couplers support several networks. Therefore, this category requires to have multiple `NetworkTypes` and multiple efficiency attributes that describe the coupling between the networks. The following attributes for resource planning are required for couplers:

- NetworkType for network 1
- NetworkType for network 2
- Installed power capacity network 1 - in [kWh]
- Installed power capacity network 2 - in [kWh]
- Efficiencies for network 1 and 2 - in [%]
- Operational costs - in [$€/_{kWh}$] or additionally also CO_2 emissions - in [$^{g}/_{kWh}$]
- ID and a readable name

A possible concrete representation is:

Name	CHP
NetworkType (network 1)	Heat ($\leq$ 80°C)
NetworkType (network 2)	Electricity (220 V)
Installed power (network 1)	43 kW
Installed power (network 2)	21 kW
Efficiency (network 1)	61%
Efficiency (network 2)	29%
OperationalCost*	5.91 $^{ct}/_{kWh}$
CO_2 emission*	0.20 $^{g}/_{kWh}$
ID	individual URI

*Note: the same cost calculation applies here as for controllable generation components

Due to the different nature of couplers, we show another typical coupler specification:

Name	Heat pump
NetworkType (network 1)	Heat ($\leq$ 45°C)
NetworkType (network 2)	Electricity (220 V)
Installed power (network 1)	25 kW
Installed power (network 2)	10 kW
Efficiency (network 1)	250%
Efficiency (network 2)	-100%
OperationalCost*	0.0 $^{ct}/_{kWh}$
CO_2 emission*	0.0 $^{g}/_{kWh}$
ID	individual URI

*Note: the operational costs and CO_2 emission of zero mean that no specific fuel costs are involved to operate the component. However, it does not mean that the component can be used without any costs at all, due to an implicit connection to the secondary network, where the negative efficiency indicate that additional electricity is required. This electricity has to be bought from the market or be produced from another component and therefore generates additional costs that are allocated to thes other component.

Following components that are used within buildings belong to the category `Coupler`:

Name	Fuel	Network 1	Network 2
CHP	gas, oil	heat/electricity	electricity/heat
Heat pump (air source)	-	heat	electricity
Heat pump (water source)	-	heat	electricity
Fuel cell	hydrogen, gas	heat/electricity	electricity/heat
Electrical heating	-	heat	electricity

Demand

Finally, the last category is `Demand`. It denotes all components and systems, whose power supply must be provided by other generation components. The required supply of entire buildings or quarters is represented with a consumption profile. Certain buildings have a rather predictable demand profile, while others, especially private households are quite difficult to obtain, due of the strong correlation with the user behaviour that is more stochastic. The forecast become more easy with a larger number of private households due to better statistics (Esslinger and Witzmann, 2012). The specification supports two options to work with demand profiles. One option is to receive a concrete demand profile, similar to forecasts of volatile units. The other option is that a building does not provide a demand profile. In this case MEMAP requires to create its own demand forecast[7]. Further, we expect that buildings offer a price information to cover demand. This is for instance the market price that a normal consumer pays. In addition, we prepare MEMAP to handle different optimization criteria, such as price and CO_2 emissions. Therefore, we require an optimization flag for the demand, which represents the desired optimization criteria. This enables to steer MEMAP's MPC towards an economical price based optimization or an ecological CO_2 optimization. In summary, the following attributes are required for demand:

- ForecastType (forecast available / forecast not available)
- Time series for demand - in [kW over time]
- Operating costs for the connection point (electricity price / heat price)
- Optimization criteria - e.g. as ENUM of the quantity price, CO_2
- NetworkType, ID and a readable name

A possible concrete representation is:

Name	Building XYZ
ForecastType	Forecast available
NetworkType	Electricity
TimeSeries	[(0:00,0.5kW),(0:15,0.6kW),(0:30,0.4kW),...]
CostReference	Electricity market
OptimizationCriteria	price
ID	individual URI

[7]Note, to obtain reliable consumption profiles is a large topic on its own. It goes beyond the scope of this thesis. In this case study we assume that a forecast profile is available. We use the approach described by (Jambagi et al., 2015; Kramer et al., 2016, 2017) to generate the electricity and heat profiles for the MEMAP case study.

Schematic input abstraction

All five input categories represent the building EMS data on an abstract level. That can also be visualized as a schematic house representation similar as in SCADA systems and appears more familiar for mechanical electrical plumbing engineers (Figure 6.4), who work with similar representations for the dimensioning of the technical equipment and economical calculations. Each component (e.g. ⟨Ctrl.Gen.⟩, which represents a controllable component) has to be instantiated with its individual parameters. This allows to model the input for MEMAP for the simulation.

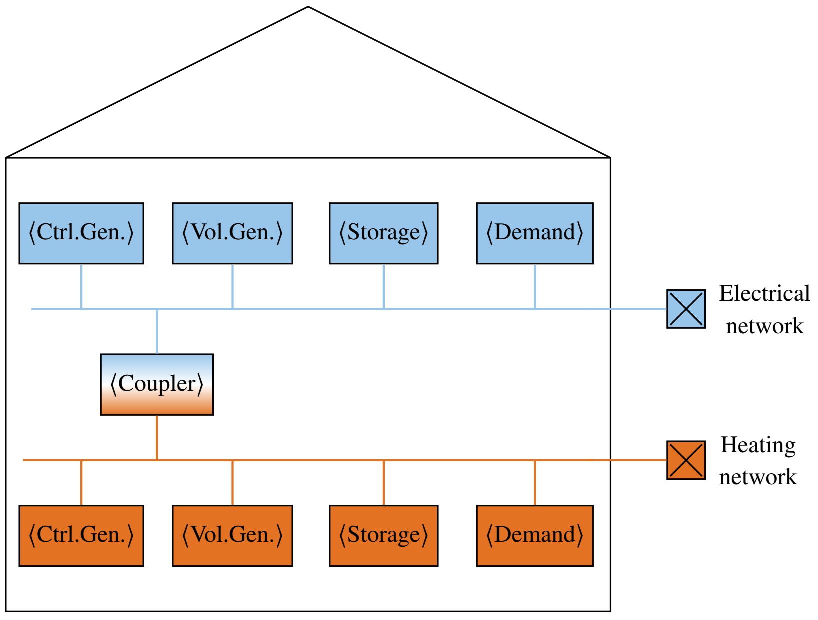


Figure 6.4: A schematic representation of MEMAP's input interface.

Planning output

The output specification of MEMAP for the planning data interface is straight forward. Every building offers a set of available options for the coordination represented as resources. The output is therefore a list of scheduled MPC set-points for each registered resource. The set-points are communicated as a time series with expected power values for certain time intervals. Every EMS receives only the set-points for its own resources, depending on the unique resource identifiers that are obtained through the input data.

Further, MEMAP provides general information as output. This includes the information about the expected costs that can be achieved as a group, and the expected costs that are achievable as an individual buildings without coordination. The same applies for the CO_2 level. This information demonstrates the potential of a coordinated approach and allows to decide for the EMS, whether it benefits from MEMAP or not. We explain further below how this is realized in MEMAP. Finally, we also leave further open data fields to extend the output interface depending on the user requirements.

Live data input and output specification

The intention of MEMAP is the support of EMS controlled buildings, so that they establish a more energy and cost efficient power supply in a group. Therefore, in addition to optimized schedules the system needs information to check, whether all EMSs operate as intended, or if an adaptation of the schedules is required. This interface extension is labelled as live data and presented in Figure 6.5. This improves the monitoring within MEMAP.

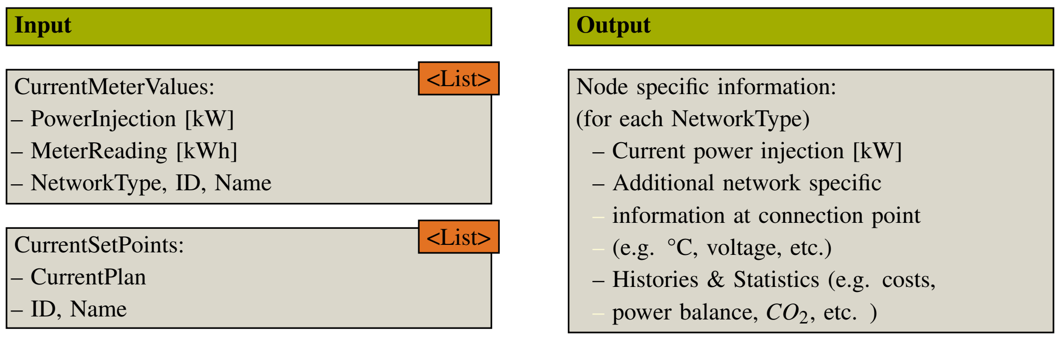

Input
<List>
CurrentMeterValues:
– PowerInjection [kW]
– MeterReading [kWh]
– NetworkType, ID, Name
<List>
CurrentSetPoints:
– CurrentPlan
– ID, Name
Output
Node specific information:
(for each NetworkType)
– Current power injection [kW]
– Additional network specific information at connection point (e.g. °C, voltage, etc.)
– Histories & Statistics (e.g. costs, power balance, CO_2, etc.)

Figure 6.5: Input/output interface specification of MEMAP towards its children: live data.

Live data input covers two categories. We require `CurrentMeterValues` as input for monitoring. It describes the building's power injection into the network. The feed in and the feed out from the network is represented as `PowerInjection`. It has the unit of [kW]. The number is positive, if the power network receives power and negative, if power is taken from it, similar as a wallet. To consider the power over time (i.e. time integrations), we also require the current `MeterReading` in [kWh]. These readings helps to monitor the fit to the schedules. All current meter values measure a dedicated `NetworkType`.

The next category `CurrentSetPoints` allows to check whether the EMS systems have agreed on the schedules and plan their resources accordingly. The agreement is checked with the `CurrentPlan`, which should correspond to the previously received MPC signal. Further each plan has the ID of the corresponding resource and a human readable name.

In the process of designing the interface for the live data input, we have thought about adding more data to enable additional control options. Such additional control options help MEMAP to react on occurring deviations in the system. The need for such short time corrections will definitely occur in real systems, either due to deviations from the forecasts, or due to other unpredictable situations, such as failures or clouds that covers photovoltaic systems. Such additional data points also open new control and specification questions for MEMAP and increase the complexity of the composed system. The questions would be similar as described with the *flexibilities* in the previous VPP case study in Chapter 5. Due to the length of this chapter, and to avoid reoccurring content, we have skipped this specification here.

Finally, the **live data output** of MEMAP addresses each EMS system as an individual node. The output data consists of current values that MEMAP observes at the connection points of each EMS. This includes the observed power injections, and specific network information such as the temperature of the input and the return flow, network pressure or electrical voltage. In addition to the current state, the each EMS system is interested in historical and statistical information, which shows how the building performed during the last days, months or years. The transparency is particularly required for the economical costs, but it also helps the EMS operators and MEP engineers in their own decision making, such as further investments or maintenance activities. The output interface of MEMAP for the general status and the node specific information can be used to develop front-ends and reporting for the users. This can be done either in the platform itself, or within the individual building EMS, depending on the endeavour of the EMS provider.

Coordination behaviour and the creation of the MPC signal

After the specification of the interface we define the coordination behaviour of MEMAP. Note, MEMAP is an EMS in the role of a coordination system. The coordination behaviour is a function that uses its input, thus, the data from the input interface, to create an output. The specification of the behaviour is essentially the MPC that we introduced in Step 1. Here we describe the model composition from the input data so that the MPC can be carried out.

MEMAP interacts with several building EMS. Each EMS has the specified interface from which MEMAP receives data and sends data to it. It provides five lists with different resource types. Each list might have one or more resources, but the list can also be empty. Therefore the first step of MEMAP's behaviour is to construct the optimization model, i.e. the matrices from equations 6.11 to 6.17 in order to solve it in a second step, i.e. equations 6.1. This is straight forward. First, we compose the demand vector using all demand resources from the buildings. Then, we sum the demand for each time step and create an **b**-vector of length $2 \cdot N_H$ for a two network system and with N_H being the MPC horizon. After that we compose the remaining resources. Each resource is used to create its own sub-matrix as explained before (eq. 6.11-6.17) and the corresponding constraint and price vectors $\mathbf{x_{lb}}, \mathbf{x_{ub}}, \mathbf{h}, \boldsymbol{\lambda}$. Then, if an external network is available, we add it to consider the buying of external power as well. Finally, all matrices are composed. We exemplary show the composed matrix **A** is in Figure 6.6, where we include several concrete component to illustrate their individual sub-matrices. The diagonal elements are marked in blue. Note, that the MPC is very modular. Many further resources can be added easily. Additional networks can be considered as well.

After this, MEMAP calculates the MPC signal for the given time horizon. The solution yields the **x** vector. It represents the desired power injection schedule for the generation and storage components. The **x** vector corresponds to the MPC set-points for the individual resources. Therefore, the last behaviour step of MEMAP is to decompose the **x**-vector in its individual components based on the IDs of the resources, calculate the individual set-points using the resource efficiencies and send those to the related EMS. Further, for the district heating we introduced in equation 3.25 (cf. Section 3.1) that the heat power has two variable parameters on the district heat side, i.e. $\dot{m}_{pr.}$ and $\Delta T_{pr.}$. To keep the hydraulics stable (i.e. the mass flow rates for the cold and warm side must be balanced) the set-points assume that $\Delta T_{pr.}$ is kept constant. In this case the desired power injections of each building (equation 6.2) and the relation $P_{Q,i} \propto \dot{m}_{i,pr.}$ can be reformulated into:

$$\sum_{i=1}^{n} \dot{m}_{i,pr.}(t) = 0, \tag{6.18}$$

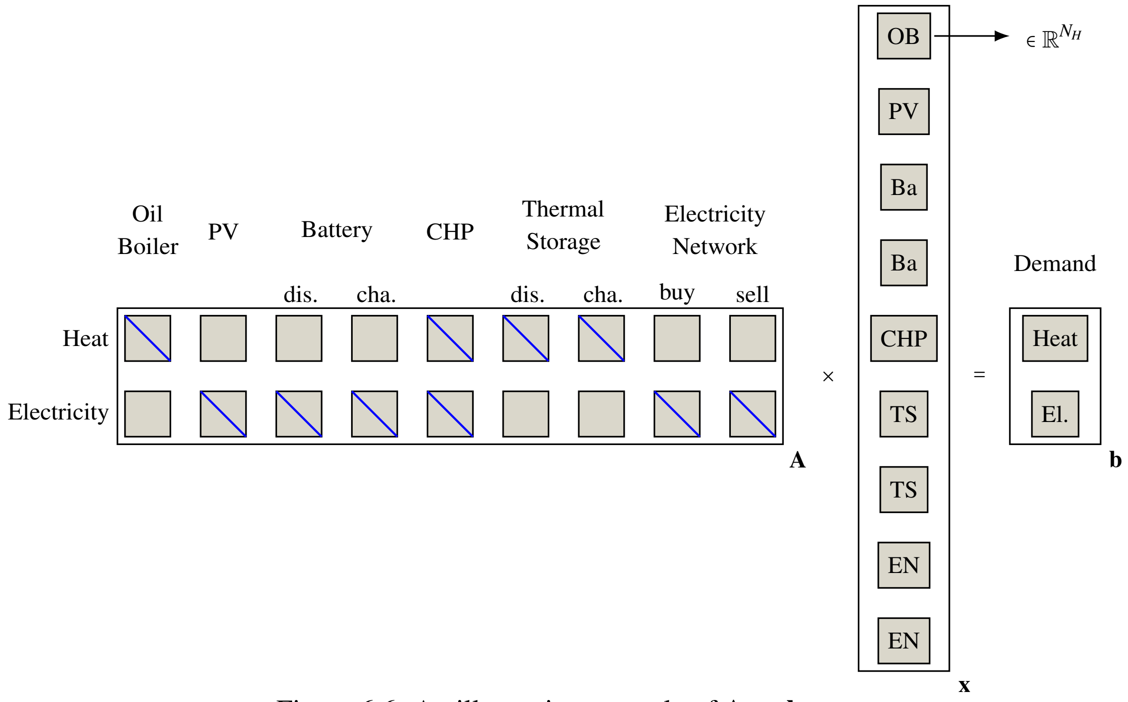


Figure 6.6: An illustrative example of $\mathbf{A}\mathbf{x} = \mathbf{b}$.

which keeps the mass flows balanced[8]. The realization of the set-points is left to the EMS. Hence, each EMS that participates in the coordination process, is expected to control its internal equipment accordingly. MEMAP monitors whether the EMS received the set-points (live data interface), monitors the related power flows and calculates new schedules if required.

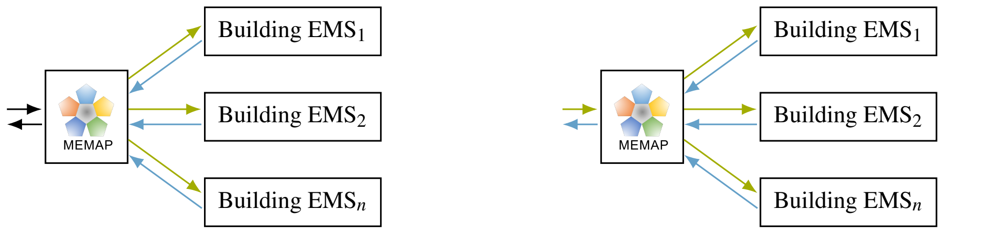
Two remarks are important here. The modular composition of the MPC using interface data leads to a very flexible system, such that modifications between different time steps are allowed. For instance, it possible to dynamically adapt the parameters of the resources after each time step, such as efficiency factors of heat pumps that depend on the environmental conditions or prices when the purchase price of gas or oil changes. Resources can be added or removed, for instance if a device is added to a building, or if a device switches to maintenance mode. Such modifications occur inside the individual EMS. MEMAP's behaviour considers them with its modular MPC.

Another remark concerns the selection of components for district heating. District heating operates at certain temperatures, e.g. see Lund et al. (2014). The temperature level determines whether a component can feed into that network or not. For instance, if the temperature is at 80°C in district heating, but a heat pump delivers only 45°C, it is not able to participate. This is also true for the demand, but with reversed conditions. A building might require to have 80°C as its heating demand, but if the district heating is operated at 45°C, then the demand cannot be supplied from that source of power. Hence, the MEMAP needs to select all suitable resources that can participate at the coordination process, i.e. being an input for the MPC. Such a selection or the possibility to have more pipes with different temperature levels is currently being researched, for instance within the CoSES Research Centre[9]. It is not in the scope of this case study.

[8]Note, imbalances in the hydraulic system might be balanced with low loss headers in district heating systems.
[9]CoSES: Center for Combined Smart Energy Systems in Garching, Germany, 2019.

6.4.3 Step 3: Specification of Aggregation and Delegation

In the previous steps, we have handled only an interaction between a central coordination system with multiple children (Figure 6.7a). For a hierarchic structure, we further have to define the aggregation and delegation behaviour (Figure 6.7b). The approach is similar as in our first case study from the previous chapter.



(a) Coordination I(↑)/O(↑). (b) Aggregation I/O (↑) and Delegation I/O (↑).

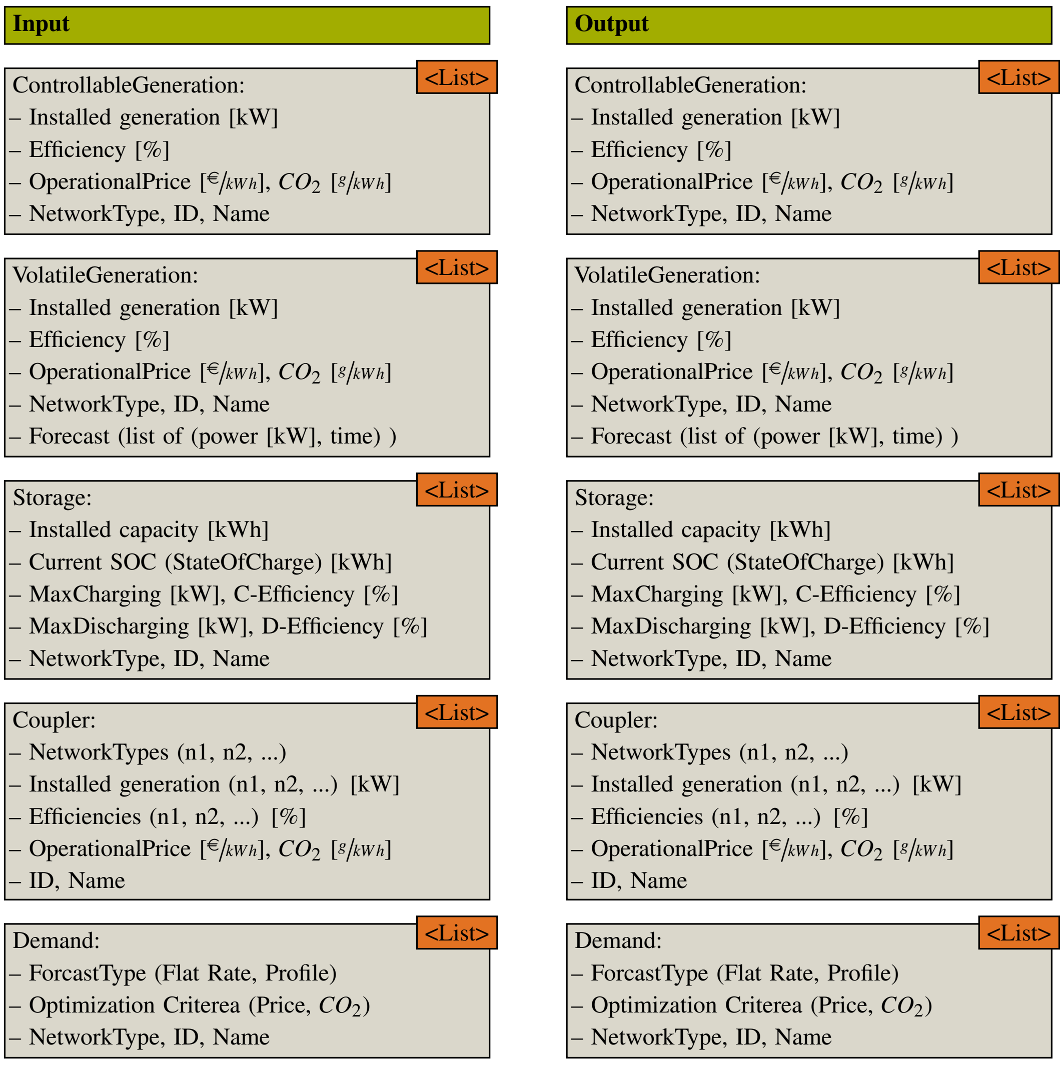
Figure 6.7: MEMAP in a hierarchical setting with the corresponding I/O interfaces.

Aggregation behaviour

We have seen that the presented coordination is already modular with generation units, demand profiles, or storage components. A higher amount of resources increases the optimization effort quite significantly. To overcome this issue, the idea in a hierarchy is not to optimize individual buildings, but as aggregated systems representing quarters, while still being able to use the same coordination approaches, i.e. the same mathematical model (equations 6.1) as before.

The aggregation behaviour, thus the *C5-case*, is a mapping from MEMAP's previously defined input interface to a similarly defined output interface. The idea is that MEMAP is able to provide the same resource options to a larger coordination system in the same way as individual building EMS. This realizes a hierarchic structure that is investigated with prosumer oriented architectures or the cellular approach, as presented in Chapter 1. The *C5-case* demonstrates the necessary system boundaries from the software perspective with respect to higher levels. The *C5-case* interface is shown in Figure 6.8. We explain the aggregation behaviour in the following based on the individual resource types.

`ControllableGeneration` components differ by their installed generation limits, efficiency, costs and of course the network. The dominant criteria for scheduling controllable resources is the price. Hence, we start the aggregation by normalizing all costs to a nominal efficiency of 100%. This means that a component with an efficiency of 80% and a cost of 6 $^{cent}/_{kWh}$ is normalized to an efficiency of 1 and a cost of 7.5 $^{cent}/_{kWh}$. If two normalized costs are close and have the same `NetworkType`, aggregation is possible. The interpretation of "close" is of course rather flexible. To demonstrate one possible aggregation example, we create two groups, one group that represents low cost components and one the high cost components. The groups are divided by the median price. The aggregation combines two controllable components by adding their `InstalledPower`, calculate the new cost (weighted average) and giving the aggregated resource a new ID and a name. We keep track of the aggregated components with lists that represent the constituents of the new resource.



Input

ControllableGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– NetworkType, ID, Name

VolatileGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– NetworkType, ID, Name
– Forecast (list of (power [kW], time))

Storage: <List>
– Installed capacity [kWh]
– Current SOC (StateOfCharge) [kWh]
– MaxCharging [kW], C-Efficiency [%]
– MaxDischarging [kW], D-Efficiency [%]
– NetworkType, ID, Name

Coupler: <List>
– NetworkTypes (n1, n2, ...)
– Installed generation (n1, n2, ...) [kW]
– Efficiencies (n1, n2, ...) [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– ID, Name

Demand: <List>
– ForcastType (Flat Rate, Profile)
– Optimization Criterea (Price, CO_2)
– NetworkType, ID, Name

Output

ControllableGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– NetworkType, ID, Name

VolatileGeneration: <List>
– Installed generation [kW]
– Efficiency [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– NetworkType, ID, Name
– Forecast (list of (power [kW], time))

Storage: <List>
– Installed capacity [kWh]
– Current SOC (StateOfCharge) [kWh]
– MaxCharging [kW], C-Efficiency [%]
– MaxDischarging [kW], D-Efficiency [%]
– NetworkType, ID, Name

Coupler: <List>
– NetworkTypes (n1, n2, ...)
– Installed generation (n1, n2, ...) [kW]
– Efficiencies (n1, n2, ...) [%]
– OperationalPrice [$^{€}/_{kWh}$], CO_2 [$^{g}/_{kWh}$]
– ID, Name

Demand: <List>
– ForcastType (Flat Rate, Profile)
– Optimization Criterea (Price, CO_2)
– NetworkType, ID, Name

Figure 6.8: Interface specification of MEMAP towards higher level EMS.

The aggregation of `VolatileGeneration` is easier, since the operational costs are usually close to zero. Hence, all volatile generation components can be aggregated to a single volatile resource. We add all parameters and forecasts that are later reflected in the upper limits of the coordination model.

The aggregation of `Coupler` components is more complex, due to the large variety of efficiencies, costs, installed power, COPs, etc. Thus, we cannot reuse the aggregation mechanisms for `ControllableGeneration`, particularly because the combination of the two efficiencies (e.g. CHP and heat pumps) leads to different energy injections on the networks and they also have a significant difference in their expected temperature range. Therefore, we use a heuristic that differentiates between two typical coupler components, the heat pump and the CHP in the first step[10]. In the second step we aggregate CHPs and heat pumps separately.

[10]Note that we describe only one possible aggregation example that we can use for our case study. But there might be other good aggregation solutions to reduce the optimization effort as well.

The first aggregation applies to components without fuel costs, such as heat pumps. They provide power to one network by taking power from another one. The efficiency is represented by the COP factor for the primary network. A higher COP means a better conversion. Therefore, we first sort heat pumps by their COP. Then, we aggregate them according their efficiencies using two groups (low COP, high COP) as described before. The aggregation sums up the static attributes, calculates a new COP, gives the new resource an ID and a new name. It then uses two lists to keep track of the group composition.

The second aggregation applies to components with fuel costs, such as CHPs. Here, we calculate the normalized costs first, where we consider the efficiency for the heat network. The reason is that CHPs are primarily used for heat generation. Then, we sort the normalized costs, create two sub-groups with components that have similar costs separated by the median value, calculate new average cost for each sub-group and sum up the operational limits as before. Finally, we give both resources a new ID, a new name, and keep two lists for the group composition. The different categorizations that are used for the aggregation of couplers are sketched in Figure 6.9.

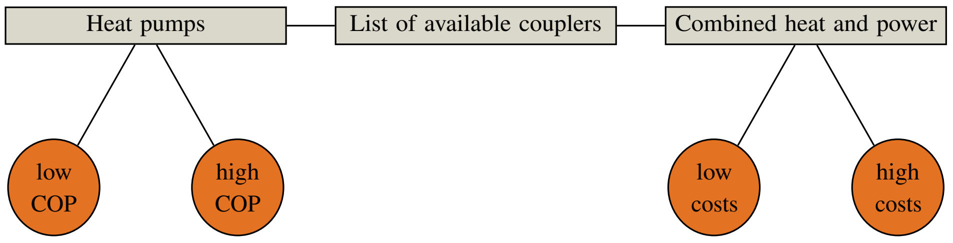


Figure 6.9: Aggregation structure of `Coupler` components.

Further, the aggregation of `Storage` requires no sorting of costs. Instead we have to take care about different speed rates for charging and discharging. For instance, if we have one storage component that allows fast charging, but has a comparatively little capacity, for instance a 10 kW charging rate and 10 kWh capacity, and we have another storage component that has moderate charging rate and a high capacity, for instance 3 kW charging rate and 30 kWh capacity, an aggregation that has a 13 kWh charging rate and 40 kWh capacity is not a good choice. Therefore, we aggregate storage components based on their network type and the time to completely charge and discharge the storage component. Luckily, efficiencies for storages are usually close to 100%. Nevertheless, for the aggregation they are combined as a weighted average as well. Note, with respect to the SOC (0..100%), when aggregation is active, we should try to balance our aggregated storage components keeping the SOC at a comparable level between those storage components that are aggregated into one resource.

Finally, the aggregation of the `Demand` is a sum. This is anyway done when the vector **b** is created. It can be directly communicated upwards for the corresponding network types.

As a short summary, aggregation is used to improve the scalability of the optimization, by reducing the number of components that are used in the optimization. We group and aggregate similar components to obtain similar coordination results. Of course, such an aggregation reduces the level of details and might slightly blurs the potential operation. Nevertheless, the important aspect is that more efficient components are differentiable from the less efficient ones, which allows to prefer the more efficient ones for the operation. This distinction is important to achieve a more efficient system.

Delegation behaviour

After the aggregation behaviour we discuss also the other direction, namely the *P1-case* delegation that describes the mapping of set-points for the aggregated resources to its original representatives. The output specification in Figure 6.3 shows that decomposition applies to the set-points of the MPC or to the general information. The coordination aims to reduce the overall costs or emissions with an optimized scheduling. We therefore specify a delegation behaviour only for the MPC signals for the five component types, not the general information.

`ControllableGeneration`'s MPC signal corresponds to a list of set-points valid at specific time intervals. The task of the delegation is to create set-points for the children EMS in such way, that the aggregated set-points are fulfilled for each time step. It is straight forward for the boundary cases with 0% and 100% that correspond to the maximum production or turned off devices. In this case delegation forwards the values, accordingly. For the intermediate case, the MPC set-points have to be adjusted for each resource. For the best efficiency the signal is delegated to the cheapest resources first. However, this might lead to sharp production ramps for individual devices. For a smoother device operation the set-points can be chosen proportionally.

The delegation of the MPC set-point for `VolatileGeneration` corresponds to the task of matching the the MPC set-points with available production profiles from the children. Since the MPC signal is created from the previously communicated forecasts for volatile generation ($\mathbf{x_{lb}}$ and $\mathbf{x_{ub}}$), it has to be mapped back to the corresponding components that were able to produce the required amount. Usually, volatile generation is used at full capacity since the operational costs are almost zero. The delegation therefore forwards that 100% signal to all volatile components. Only if explicit limitation of the production is required (for instance for use cases like congestion control) the delegation proportionally limits all of the children's production to reach a fair utilization of the components.

While the aggregation of `Coupler` components is complex, the delegation is not that difficult and similar to the delegation of `ControllableGeneration`. The boundary cases with 0% and 100% are straight forward. The intermediate cases require to send the proportional signal to each component to keep a good quality of the MPC signal. To improve costs, we can use more efficient components first (lower normalized costs or a higher COP) but we need to decide on one network, i.e. heating to keep the hydraulic network more stable. This results in small deviations in the electricity schedule, due to the fact, that each coupler has two efficiency values for each networks. However, since we use the more efficient components, we either save heat costs (for CHPs) or can sell not required electricity (heat pumps). Nonetheless, we have to be careful and decide, if our delegation prefers a higher precision or small deviations but slightly improved costs.

`Storage` components' MPC signal includes the charging and discharging rates for the time steps. The efficiency is the average weighted efficiency. For delegation we proportionally allocate the set-points of the aggregated (virtual) storage based on the charging rates of the sub-ordinate storage components, to keep a good MPC quality, similarly as a battery management system.

The `Demand` is an expected profile that is part of the optimization problem formulation (see vector $\mathbf{b}$). No MPC signal is required, therefore, nothing has to be delegated. Note, we specifically decided not to consider demand response use cases in this thesis. Otherwise, we the delegation of demand response would be an interesting research topic on its own.

6.4.4 Step 4: Implementation, Design Patterns and Industrial Interfaces

In this part, we present the implementation of our coordination approach. This allows us to evaluate its feasibility and obtain quantitative results with respect to the expected potential improvement of the energy costs and savings within interconnected EMS. The implementation of the case study is carried out with SESSIM, the co-simulation framework that is developed for this thesis and introduced in Section 4.3. The co-simulation framework implements every EMS, such as an EMS of a building or a higher level coordination system, as an independent actor. Each actor has its own environment such that its internal data is not accessible from outside, thus, internal data is not exposed. The interaction between different actors is only possible via communication with messages. We use only serializable data as message contents to represent the data exchange. This approach allows us to generate different interfaces representations directly from those data points. With that SESSIM becomes and excellent framework to study interactive systems and specify the necessary interfaces and system behaviour.

The implementation of the MEMAP case study includes two steps. The first step is the representation of all participating EMS (every building and the coordination system) completely within the SESSIM framework and the implementation of the systems behaviour as an actor logic. This allows to simulate and study different scenarios. For the first step we only apply the *composite pattern* as a central architecture design pattern, since we are interested in a scalable simulation without clients and servers. The second step is the extension of the framework into a server based platform that interacts with real EMS systems. It extends the first step by adding *clients and servers*, *layers*, *unified interfaces* and *publish-subscribe* mechanism to create a fully operational, distributed system. In this step, MEMAP connects to various EMS interfaces of our industrial partners, such as the SAUTER Vision System[11] (see also Figure 6.1). After that it reads the data from those interfaces, performs exactly the same calculations as within the simulation environment and sends the MPC signals to the remote systems. This helps to test the logic in a simulation first and then evaluate the interaction with real hardware systems and with that provide a hardware in the loop test environment to evaluate control models later on. The implementation is available as open source[12]. We describe our two steps in the following.

Implementation as a co-simulation (tool for planning)

The first step of the implementation was to create a simulation environment, in which we are able to define our data points and the behaviour of different systems including, devices, EMS and MEMAP. We used our co-simulation framework. In SESSIM, the central element for the implementation is an actor. An actor receives messages, executes a behaviour and sends messages. For the case study we created a hierarchical system of interacting actors (see Figure 6.10). We developed three different types of actors: `Device` actors that represent the devices within a building, `Building` actors that control the devices and represents local EMS and `Aggregator` actors that coordinate the individual EMS and represent the MEMAP coordination system.

[11] A commercial building management system that is used for large and complex systems, like quarters, including a variety of different devices, sensors and actuators for control.

[12] SESSIM: `https://github.com/SES-fortiss/SmartGridCoSimulation`, last accessed in March 2020

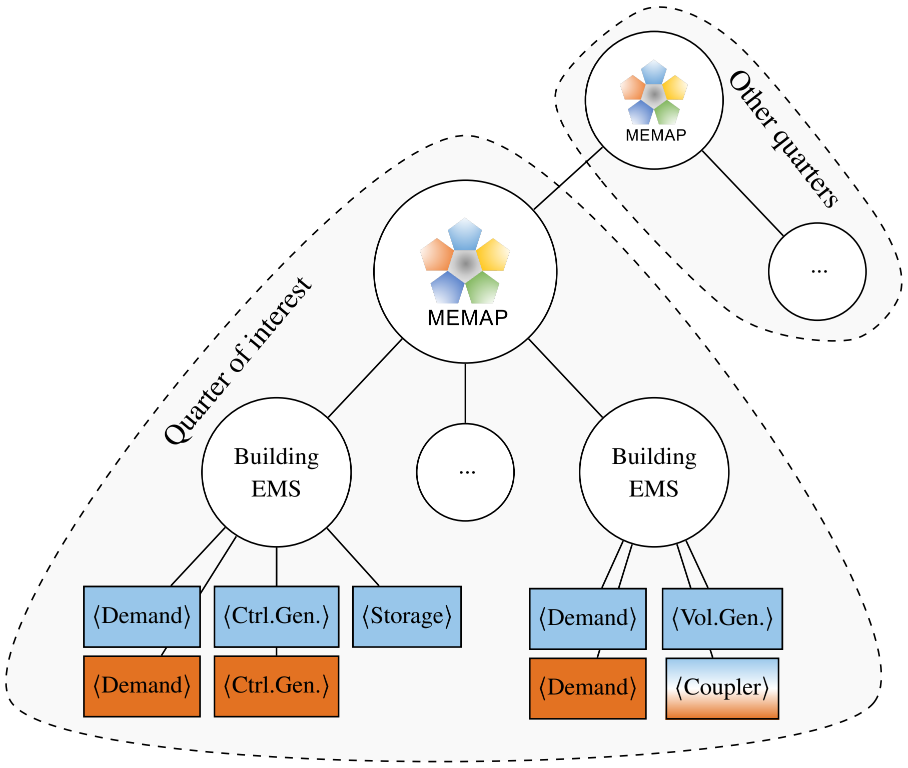


Figure 6.10: Chosen actor hierarchy for the implementation in the SESSIM framework.

The type of the actor determines its interface and behaviour. The `Device` actor has behaviour (and interface) that represents our five basic resources. It sends its device parameters to the building EMS. When its gets a response, it executes the requested set-point. As EMS systems, the `Aggregator` actor (MEMAP) and `Building` actors (B-EMS) are almost equal. They have the same external interfaces and behaviour. Only the possibility to have specific local devices differentiates them. They receive lists of resource messages from below and are able to carry out the MPC that was presented above, with the difference that the EMS actors receive individual device resource messages from their internally available components, while the MEMAP actors receive messages that contain lists of resources, as specified beforehand.

Consequently, there are only two different message types being exchanged between the EMS and MEMAP coordination system in SESSIM. One message type is called `BuildingMessage`. It communicates resources upwards and represents the input data of MEMAP as described above in Figure 6.3. The second message contains the content for the communication downwards. It is called `OptimizationResult` and contains the desired set points for the individual IDs of the communicated energy resources. It represents the output data of MEMAP (see Figure 6.5).

To implement the behaviour we use the main methods of the `SessimActor`'s behaviour steps that we introduced in the general description of the SESSIM framework in Figure 4.15. It is convenient to use the following two methods:

1. `handleRequest()`: This method triggers, when a request from the parent actor arrives. It adapts the EMS internal state based on the request. If child components are available, own requests to the children are prepared. After this, the requests are sent downwards.

2. `makeDecision()`: This method triggers, when all answers from the children arrive. It processes the messages from the children, runs the optimization and adapts the internal state. It also prepares an own answer to the parent. Afterwards, an answer message is sent upwards to the higher EMS coordination system.

The first step represents the mapping of the input to the output from the higher hierarchies to the lower ones. The second step represents the input to output mapping for the other direction. The splitting of the behaviour in different sub-steps is helpful to implement different functionalities in MEMAP. We explain in the following how our behaviour is implemented.

makeDecision()

This method implements the behaviour of an EMS actor that processes the input from the children. In our case study implementation this is the method, where the aggregation and optimization takes place. When the EMS actors receive the resource data from their children, they create new `BuildingMessage`s that are sent upwards to their parents. The parent actors receive several `BuildingMessages` from their EMS children. They use those data as an input for the MPC. Every instance of the `BuildingMessage` is added to the matrix representations (equations 6.11 - 6.17), as we described in the previous section. After all instances are added, we solve the optimization problem using the `joptimizer` library[13]. The optimization solves equations 6.1 in the next step. The solution vector is stored together with the names in a `Map`, which maps the identifiers of the EMS resources of type `String` to an MPC signal represented as `double[]` arrays. This map represents the MPC schedule. It is communicated as request during the next simulation time step within our simulation.

Further, this method allows to implement additional functionality. First it is used to store the decision results of each time step to generate a history of decisions. Two options have been implemented and tested to store the history. The simple method is a CSV file writer, which updates a result file with the current information. For more advanced functionalities we initialize a data base connection to store the information.

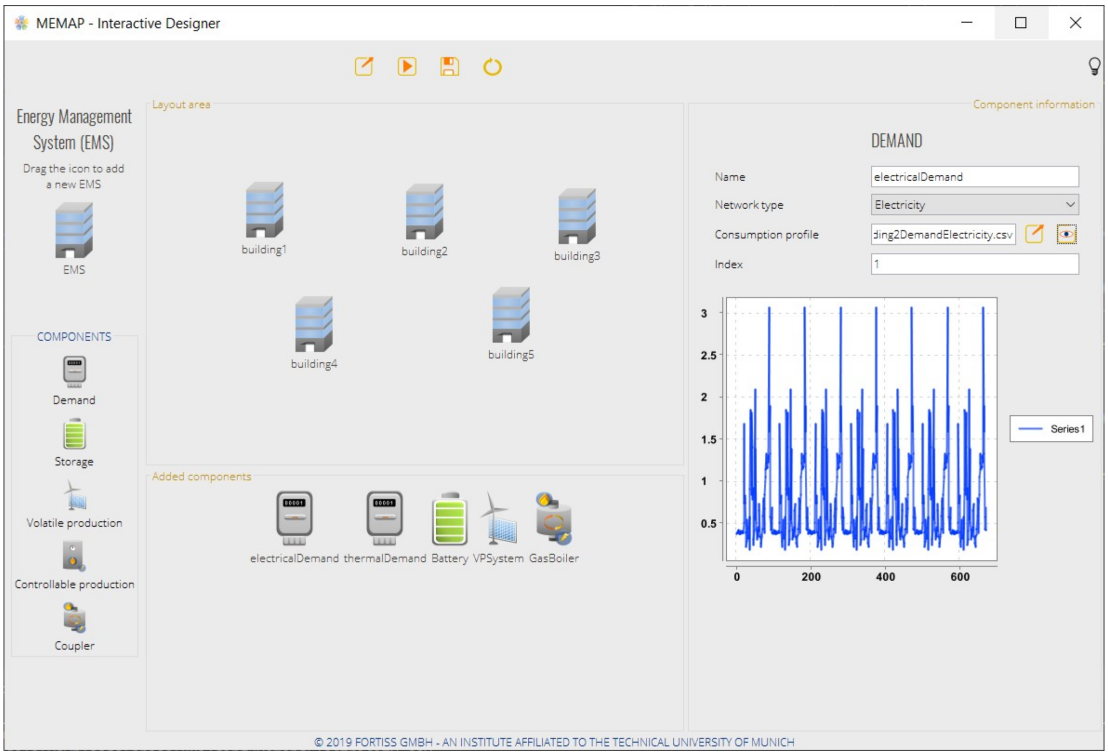
handleRequest()

The second important method of the `SessimActor`'s behaviour processes the input from the parent. This input contains the MPC set-points from the parent. The actor reads these set-points and delegates them to the lower hierarchic components in the simulation, i.e. a building communicates those set-points to its devices. It also updates the live data (see Figure 6.5). Note, since the EMS system is quite complex and has to check additional internal constrains, is can decide itself whether the component fulfils the schedule or not. The parent has no access to the components below the EMS system, thus, the devices within a building. This is an important constraint by our industrial project partners.

[13]An open source optimization library implemented in java: `http://www.joptimizer.com/`. The library is based on the `commons-math` and the `colt` libraries, which we also used in our framework for the Newton-Raphson solver, that was presented in section 4.3

Configuration of the simulation

The configuration of SESSIM is used to model the scenarios that are of interest. SESSIM provides a single class for that task to keep the maintainability as simple as possible, the so-called `Topology`. The topology configures the framework and adds the desired actors to the system. Actors can be added by giving them a name and their parent actor. Power component actors additionally need parameters, similarly as shown in the specification of input data fields in Figure 6.3, and if required a specification of the profile reader, which is responsible to obtain the desired demand profile or the photovoltaic forecast. The `Topology` configuration class also specifies additional simulation parameters, like the number of simulation steps, their time intervals, the MPC horizon length, etc. The configuration class is also extractable to an external file, so that we can configure the simulation in a JSON like file format. This enables also to create graphical user interfaces for the configuration of MEMAP, as shown in Figure 6.11.



Name: electricalDemand
Network type: Electricity
Consumption profile: ding2DemandElectricity.csv
Index: 1

Figure 6.11: A visualisation to configure the simulation. It is used as a planning tool in MEMAP.

Visualization of the simulation

The visualization of our case study has two parts, a live view and a view for time series analysis. The live view displays the current values of the actor using a light weight `jetty` server. Every actor displays its current state as a *application/json* output, which is then used for web based visualizations implemented with JavaScript. For the generation of historical views we used two approaches, depending on the purpose. For presentations and analysis with graphs, which we show in the upcoming section, we use CSV files. For the web based applications have developed

REST based history requests, that provide the required back-end connection to a postgresql[14] data base to show additional statistical informations. The visualizations with servers were the first steps towards the utilization of SESSIM as a platform.

Implementation as Coordination Platform with Industrial Interfaces

In the following we describe further implementation activities that have been carried out on top of the first step. Since the research project MEMAP is still ongoing during the writing of this thesis, we present here the first steps that describe the basic architecture and interfaces. The contribution of this section is the transfer from to real running platform systems that communicate over reliable, industrially well tested and widely used automation technologies using the same architecture and interface specifications.

This server-based implementation activity relies on same data as in the messages within the simulation. But it converts them into real server based interfaces for the communication with external EMS systems. In contrast to the first case study (see Chapter 5), where we implemented the communication using REST, the idea of MEMAP is to use interfaces from the automation domain. We therefore decided to apply our architecture design pattern using OPC UA (OPC Unified Architecture) for the machine to machine communication. Most of our industrial partners have a good experience with OPC UA implementations and plan further extensions based on this interface. Therefore, the implementation of OPC UA communication technology is a promising choice and has a number of advantages. We describe our implementation and the role of the architecture design patterns in the following.

OPC UA

OPC UA is an open standard for the communication in industrial automation. It specifies the information exchange and the communication model for machine to machine interaction. Today, it is often used for the communication between devices within an automation system, between several automation systems and between automation and operational systems. OPC UA is based on clients and servers as interacting partners. Every client may interact concurrently with one or multiple servers and vice versa (OPCFoundation, 2017). Further, each system may contain multiple clients and multiple servers. OPC UA supports the conventional client server request and response pattern and publish-subscribe mechanisms. The publish-subscribe pattern is established, when a client requests the OPC UA server to subscribe to a particular data node. After subscription, the server notifies the client when the data point is updated. This reduces the communication overhead and allows to implement event based systems. Systems with OPC UA are well suited to operate in closed networks but also over the internet, since security is an inherent part of the protocol including authentication, access control and encryption. The OPC UA stack supports additionally several defined concepts, including discovery and browsing, the ability to represent structure, behaviour and semantics with type definitions, relations between nodes and access over the address space in an information model (Lange et al., 2010; OPCFoundation, 2017). This makes OPC UA well prepared for a wide range of problem domains.

[14] `https://www.postgresql.org/`, last accessed in March 2020

MEMAP's OPC UA interface

For our case study we implemented OPC UA as the main interface for the MEMAP platform. Each EMS offers its data over OPC UA to its parent, hence each EMS is an individual server system. MEMAP repeats this pattern. It implements client to communicate with its children. For the communication with its parents it has a server. The architecture is illustrated in Figure 6.12.

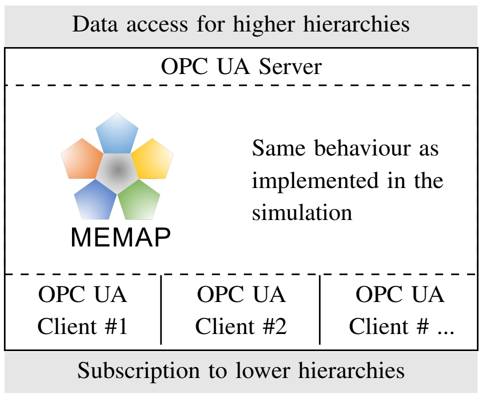

Data access for higher hierarchies
OPC UA Server
MEMAP
Same behaviour as implemented in the simulation
OPC UA Client #1
OPC UA Client #2
OPC UA Client # ...
Subscription to lower hierarchies

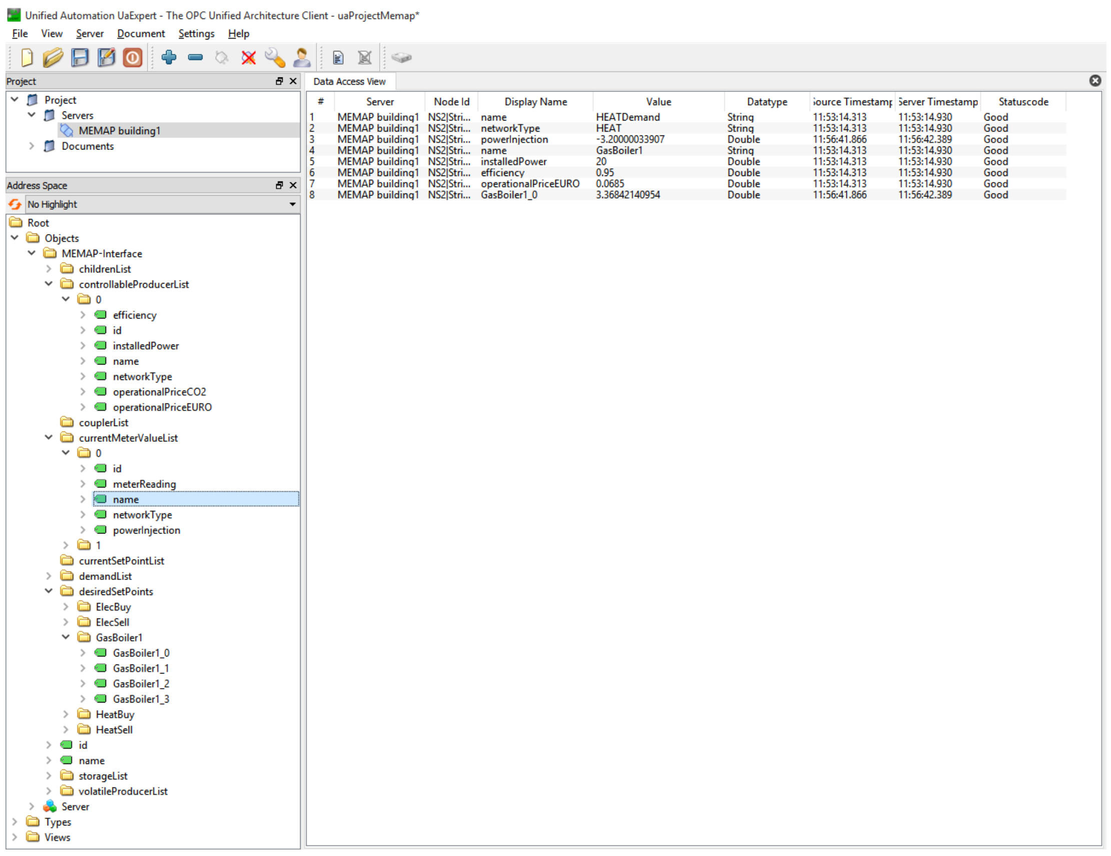
Figure 6.12: MEMAP architecture with OPC UA.

The choice for that design had several reasons. First, every EMS system already runs as a dedicated system. It provides its users the possibility to monitor and configure the system. For that it collects the data of the controllers, which it can offer to external systems as a server. Second, the benefit that the EMS implements a server is that any external system, like MEMAP, has to implement the interface of the EMS, not vice versa. This allows the EMS to offer its interface to different vendors over its API. The EMS is also controls the security settings. The third benefit is that the coordination system is aware whether the EMS follows the coordination suggestions or not. This allows to coordinate with more confidence. When an EMS is not following the schedule, the coordination system is able to adapt its planning accordingly.

The implementation of OPC UA servers and clients is realized with the `Eclipse Milo` open-source library[15]. To represent exactly the same information as in our SESSIM co-simulation, the configuration of the OPC UA server is automated. We presented that in Bytschkow et al. (2019). For generation of the interface we convert the specified `BuildingMessage` object into a OPC UA interface using a generic JSON serialization. This is achieved with the `gson` library. The interface set-up is implemented in three steps:

1. Create a JSON string with `gson`.
2. Use the string as a stream input for the `JsonReader` and use the tokens to generate two types of nodes: a) `ObjectNodes` that represent either JSON objects or JSON arrays that contain other nodes; and b) `ValueNodes`, which represent variables with specific values, such as doubles, integers or strings.
3. Configure the OPC UA server such, that all `ObjectNodes` are represented as OPC UA folders, and `ValueNodes` are represented as subscribable values.

[15]`https://github.com/eclipse/milo`, last accessed in March 2020


Unified Automation UaExpert - The OPC Unified Architecture Client - uaProjectMemap*
File View Server Document Settings Help
Project
Address Space
No Highlight
Data Access View

```
Project
  Servers
    MEMAP building1
  Documents
```

```
Root
  Objects
    MEMAP-Interface
      childrenList
      controllableProducerList
        0
          efficiency
          id
          installedPower
          name
          networkType
          operationalPriceCO2
          operationalPriceEURO
      couplerList
      currentMeterValueList
        0
          id
          meterReading
          name
          networkType
          powerInjection
        1
      currentSetPointList
      demandList
      desiredSetPoints
        ElecBuy
        ElecSell
        GasBoiler1
          GasBoiler1_0
          GasBoiler1_1
          GasBoiler1_2
          GasBoiler1_3
        HeatBuy
        HeatSell
      id
      name
      storageList
      volatileProducerList
    Server
  Types
  Views
```

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	MEMAP building1	NS2|Stri...	name	HEATDemand	String	11:53:14.313	11:53:14.930	Good
2	MEMAP building1	NS2|Stri...	networkType	HEAT	String	11:53:14.313	11:53:14.930	Good
3	MEMAP building1	NS2|Stri...	powerInjection	-3.20000033907	Double	11:56:41.866	11:56:42.389	Good
4	MEMAP building1	NS2|Stri...	name	GasBoiler1	String	11:53:14.313	11:53:14.930	Good
5	MEMAP building1	NS2|Stri...	installedPower	20	Double	11:53:14.313	11:53:14.930	Good
6	MEMAP building1	NS2|Stri...	efficiency	0.95	Double	11:53:14.313	11:53:14.930	Good
7	MEMAP building1	NS2|Stri...	operationalPriceEURO	0.0685	Double	11:53:14.313	11:53:14.930	Good
8	MEMAP building1	NS2|Stri...	GasBoiler1_0	3.36842140954	Double	11:56:41.866	11:56:42.389	Good

Figure 6.13: Snapshot of the MEMAP OPC UA interface with UaExpert[16].

The automated generation of the OPC UA server configuration creates the desired interface. We display the outcome with an industrial client software in Figure 6.13. It represents the Json `Objects` and arrays as folders. All other JSON data types are represented as `ValueNodes`, hence the data (green tags) for which the client subscribes to. The input interface from the specification (Figure 6.13) is restricted with the access level “CurrentRead”. The output interface is represented by the folder `desiredSetPoints`. It collects the set-points (the MPC-signal) with an component identifier as an array of desired values for the next timestamps. It has the access level “CurrentWrite”. This allows MEMAP to write on the server interface, while the server can continuously update its values for the input interface.

Further extension to create the platform

Finally, we describe the last extension steps of MEMAP from a simulation to a real platform running on a dedicated server environment. To test the system we first implemented that each building actors starts its own OPC UA server with the explained interface. This represents individual EMS systems. The OPC UA servers allow to subscribe to their interface values. Additionally, we have dedicated EMS systems from our industrial partners, which can be connected as well.

[16]UaExpert is an industrial OPC Client Software from Unified Automation GmbH.

Then the platform starts as second step. MEMAP offers a web interface, where we can add the endpoint *urls* of the OPC UA servers from the individual EMS. When an *url* address is provided, MEMAP starts an OPC UA client within the actor framework, that subscribes to the servers. The clients read the values of the EMS systems and set the desired set points. Additionally, every MEMAP actor can have a data base connection, that is used to store the received EMS values. The data base allows the visualization of historical data, that includes the consumption for heat and electricity, as well as the production and consumption values of the related components. This visualization of MEMAP requires one additional server. We implemented one additional jetty server for this purpose, but could have used any other server system as well.

The behaviour logic of MEMAP in the `handleRequest()` and `makeDecision()` methods remains unchanged compared to the co-simulation. That ensures that the developed algorithms are the same for the server system. But we extend the actor's behaviour of MEMAP from the co-simulation with additional methods that are responsible to trigger the process as described in Figure 4.15, since we do not purely rely on the internal communication by actors, but mainly on signals from external systems. The extensions represent the receiving process, which takes place before `handleRequest()` as messages from the parent and `makeDecision()` methods as messages from the children. In total we implement the remaining methods from Figure 4.15.

The first extension is required to establish the top-down communication path. It begins by reading the parents requests in OPC UA. For this, we implement a trigger to read the `desiredSetPoint`. This event represents that the new set-points are provided by the parent and the `handleRequest()` method needs to be is executed. Without children, the `handleRequest()` method correspond to an update the desired MPC set-points internally. With children this method delegates the MPC signal received by the server to the clients (see also Figure 6.12). The clients send then own signals to their connected servers in the follow up step.

The second extension is required to establish the bottom-up communication path. In our implementation this corresponds to a read the currently available values in the EMS below using clients that are connected to the children's server interfaces. We decided with our partners on a periodic clock timer. When the periodic timer triggers, all required values are collected from the clients and the `makeDecision()` method is executed. If it has no parent above, the method performs the presented optimization, and updates the MPC results with the connected clients. Otherwise, this method aggregates the values, to update its own OPC UA server interface.

This extension implements the system according our architecture design patterns. OPC UA requires the *client-server* pattern. Further, the *composite pattern* follows our essential specification activity with an interface that offers a good abstraction from technical details of energy components such as generation units, storages and demand. It enables to define a coordination behaviour that is not only used for one particular coordination system, but also for different hierarchical levels, since our optimization handles differently equipped buildings, without the requirement for any adaptation, and can even dynamically be adapted to a changing number of available children. This is clearly beneficial for the system design. Further, the OPC UA server is only a gateway that offers data and receives data. The internal logic of the EMS, particularly the details for controlling the devices is not exposed. Thus, we have a good separation of concerns and abstraction. Note, separation of concerns require to focus on details that matter (Dijkstra, 1982), while abstraction is used to hide complexity (Liskov, 1988) and therefore makes they system easier to maintain and develop. Both principles are considered a good software engineering practise. We use internal *layers* as an architecture design pattern for that purpose. Further, we consider *unified interfaces* as follows. We use OPC UA throughout the system, but we theoretically have the possibility to implement the same interface with another representation, for instance with MQTT or REST. Finally, this case study demonstrates that the *publish-subscribe* architecture design pattern is beneficial. It increases several system properties. When we use publish-subscribe with OPC UA, we reduce the communication overhead, since we communicate only the changing parameters. This reduces the necessary data exchange. The publish-subscribe pattern was also beneficial for the system stability, since communication faults are handled intrinsically by the OPC UA libraries. Hence, we can easily reconnect, restart the system or taking other appropriate measures.

6.4.5 Step 5: Evaluation Scenario

MEMAP's benefit for energy optimization and its flexible design is demonstrated in the following. We developed together with our industrial partners multiple reference scenarios. Here, we demonstrate a collaboration of five houses of different size. Each house has an individual electricity and heat demand. Additionally each house has its own individual equipment to cover these demands. MEMAP's task is the coordination of the connected buildings to evaluate the potential energy efficiency improvement in a scenario, where buildings exchange energy. For that we assume that the houses are interconnected electrically and with district heating. The improvement of the efficiency is measured with operational costs and CO_2 emissions.

We start by modelling expected demands for one arbitrary day. We use the approach described by (Jambagi et al., 2015; Kramer et al., 2016, 2017), which specifically develops individual reference profiles for building electricity and heat demand. We create two single private house holds and three multi-family houses. The profiles are illustrated in Figure 6.14.

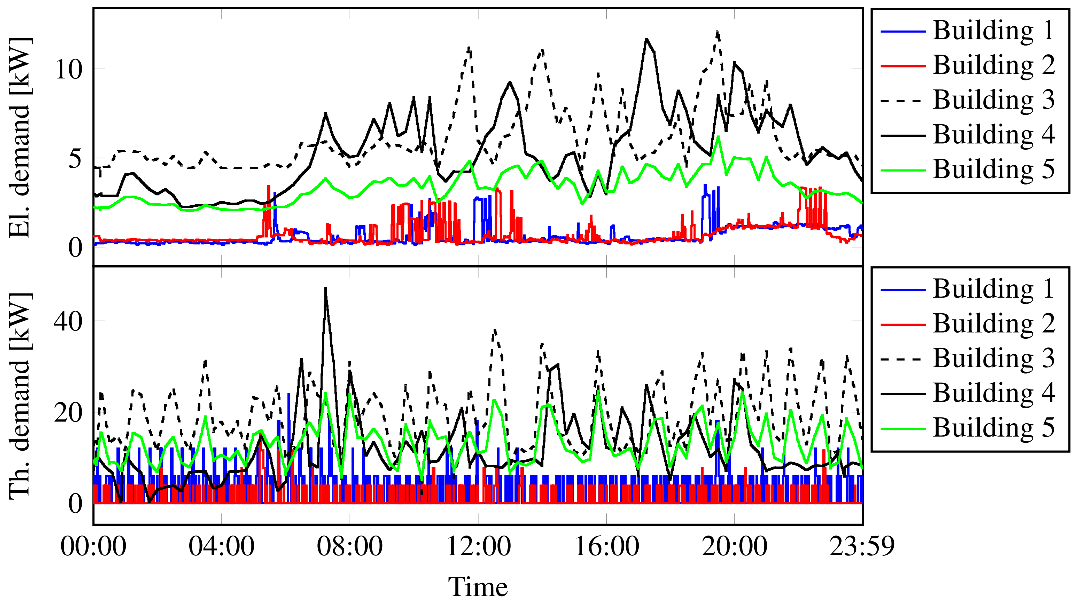


Figure 6.14: Demand profiles for the MEMAP scenario.

In the second step we model the generation of the houses to cover the demand. We use different generation units that are currently available in the market. Controllable generation units are oil and gas boilers. As volatile generation, we us photovoltaic and solar thermic components. For their profiles, we use measured values as a reference[17] to obtain solar profiles, one for a sunny day and one for cloudy day with solar in the afternoon. As storage components we use electrical batteries and thermic storage. As couplers we use CHPs and heat pumps. Each house has its individual equipment configuration, which is shown in Figure 6.15. The diversity leads to different optimization options for MEMAP, which we want to demonstrate with this case study. The five houses define a small district. For modelling the energy exchange, those five houses represent the system boundaries similar to a district shown in the beginning in Figure 3.7. Note, we would like to emphasize here that the system boundaries that we have mind when discussing the energy system perspective are different as for software systems.

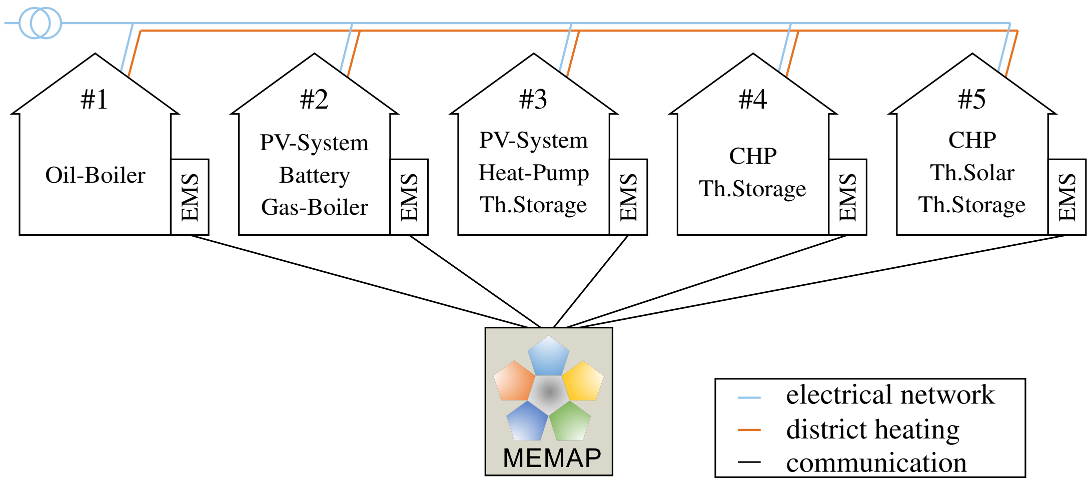


Figure 6.15: The MEMAP reference scenario.

Building #1. The first building in the scenario is a large private household with conventional equipment. It represents the most convenient and a classical set-up in Germany. The house has a daily electricity demand of 14.4 kWh, a thermal demand for heating and hot-water of 51.4 kWh (profiles as shown Figure 6.14). Electricity is supplied by the electrical network with market prices. For the supply of heat and warm water we use an oil boiler for the house. The oil boiler has a maximum installed power of 20 kW heating power and 95% efficiency. The fuel costs are assumed with 6,85 $^{ct}/_{kWh}$[18].

[17]Measured values are available from different sources. We used the demo dashboard page from discovergy: `https://discovergy.com/`, accessed in September 2018.

[18]Deutscher Energieholz- und Pellet-Verband e.V. (DEPV). Prices in Germany in August 2018. Found at the url: `https://depv.de/de/pelletpreis`, accessed in September 2018.

Building #2. The second building also represents a large private household, but it has more modern equipment to improve the costs of its supply. The demands of the house are 16.5 kWh electrical and 50.0 kWh thermal. To improve its electrical balance, it has a photovoltaic installation of 5 kWp, and a 12 kWh battery that is connected with three phases with a maximum charging and discharging rate of 9 kW[19]. The storage efficiency is assumed with 98 %, each for charging and discharging. For the heating and hot-water supply, the house has one gas boiler, also with 20 kW heating power but 98% efficiency. The fuel costs are assumed with 5.91 $^{ct}/_{kWh}$[20].

Building #3. The third building represents a medium sized multi-family household. It has a daily electrical demand of 147.8 kWh and 457 kWh for heating and hot-water. For the supply of this demand, the building has a large photovoltaic with 40 kWp and a heat pump with 10 kW electrical power and a coefficient of performance (COP) of 2.5, which is a good extension of the photovoltaic system. In addition, it has the same gas boiler as building #2 to cover the demand peaks with 20 kW heating for additional heat support and a thermal storage of 100 kWh capacity. The thermal storage can be charged and discharged with a power up to 60 kW and an efficiency of 90 %.

Building #4. The fourth building is a medium sized, multi-family building as well. Its daily demand values are 127.5 kWh electric and 286.5 kWh thermal. The demand is covered with a mini-CHP plant in combination with thermal storage. The parameters of the CHP plant are 20 kW electrical power (29% electrical efficiency) and 43 kW thermal power (61% thermal efficiency)[21]. The CHP is fuelled with gas with the costs of 5.91 $^{ct}/_{kWh}$. Building #4 has a storage with 100 kWh capacity, 60 kW (dis)charge rates and 90 % efficiency.

Building #5. Finally, the last building is a multi-family home as well, but slightly smaller than building #3 and building #4. It has a daily demand of 80.5 kWh electric and 316.9 kWh thermal. The demand is covered with a mini-CHP plant in combination with a thermal storage, similar as building #4. The CHP has the same parameters as for building #4 but slightly reduced efficiencies with 28% el. and 60% th.. Further, is has a thermal storage with the same parameters as the previous buildings. In additional, building #5 is equipped with a solar thermic system, which has an installed power of 20 kWp. For solar thermal generation we take the same profiles as for the photovoltaic production that are used for the solar of buildings #2 and #3.

Coordination assumptions. MEMAP coordinates the power exchange between houses, which is only possible if certain conditions are fulfilled by the system. This includes available hardware, measured and communicated values and no disturbing regulatory constrains. We also use some simplifications in our coordination, which we summarize in the following as assumptions.

A1 Electrical power and heat can be exchanged between the houses, i.e. there is an electrical network and district heating between the participants. Losses are neglected in the case study.

[19]The values correspond to the fenecon pro 9-12 storage system.

[20]Prices taken from Deutscher Energieholz- und Pellet-Verband e.V. as above

[21]The values correspond to the Vailland ecoPOWER 20.0 system.

A2 There is a connection to an external electrical network with market prices. If necessary, electricity can be feed out (bought) or feed in (sold) that network.

A3 Participating EMS provide reliable data, i.e. they truthfully report their demand and generation. The data is also not falsified by the communication.

A4 We do not consider electrical instabilities, such as voltage drops or frequency deviations.

A5 Components have constant efficiencies. They do not change at different set-points.

A6 Components' dynamics are not considered for planning. This means that components are allowed to change their operational state arbitrarily between time steps.

A7 Each component that produces or consumes heat interacts with the same district heating network. This also particularly assumes, that boilers, CHP, solar thermic components and heat pumps are allowed to feed into the same network.

6.4.6 Step 6: Simulation of the Scenario

The defined scenario is implemented and executed in the planning tool version of SESSIM as introduced in the previous section. To show the contribution of MEMAP we compare two situations.

- The first situation represent **individual houses**, where each house has an EMS that calculates its own optimal schedule. This is the best case that an individual EMS can achieve.
- The second situation represent the **coordination with MEMAP**, where MEMAP calculates an optimal schedule for the combination of five houses. The houses follow that schedule and exchange their energy within the quarter.

The comparison uses certain environmental conditions. For demand we use the generated data according Jambagi et al. (2015); Kramer et al. (2016, 2017). For the forecasts of solar generation, we use real historic data as introduced above. For photovoltaic components we use a "bad" summer day that generates 30% of energy compared to a summer day with excellent weather conditions[22]. For the planning, we use 15 min time intervals. Further, we compare different MPC horizon lengths n_{MPC} to evaluate its effect on the behaviour. Further, we use constant prices for the fuel and the electricity tariff of the grid. The constant electricity tariff is assumed to be 0.25 $^{€}/_{kWh}$. It represents the buying price. If power is injected back into the network, the component receives 50%. This considers that the electricity tariff (in Germany) consists around 50% from taxes and 50% for the costs of the generation, transmission and distribution. We run the simulation over the period of one week, where each day has the same demand and solar production profiles. This reduces errors due to a fixed starting condition.

All data and scenario specification are accessible online as part of the tool [23]. We published the implementation and its results in Bytschkow et al. (2019) and a follow up work together with our research partners in Heidemann et al. (2019).

[22] We used recorded data from the 21st of July 2018, which was a cloudy day with more sun in the afternoon.

[23] `https://github.com/SES-fortiss/SmartGridCoSimulation`

6.5 Results

In this section, we present the results of our scenario and discuss the effects of the coordination system. It shows the potential of combined multi energy prosumer buildings in a quarter. We demonstrate the capability to improve the current energy supply by using only the communicated data for the optimization. This gives more insight about MEMAP's functionality and the general potential to utilize energy resource more efficiently due to the interconnection.

Comparison of a quarter with and without MEMAP (fixed prices)

The first result is the demonstration that our coordination approach and the developed system has the potential to improve the energy usage for a group of buildings. The results for our scenario in terms of costs, thus, the improvement of resource utilization through MEMAP are shown in Table 6.1 for various MPC horizons. The results that describe how the generators cover the demand are shown in Figure 6.16 for building #2 (Figure 6.16). This Figure is also a good reference to explain how the MPC works in general. The other individual profiles of buildings #1, #3, #4 and #5 are presented in Appendix B, in more detail. The coordinated case with MEMAP, which has many different production components, is presented in Figure 6.17.

		Cost [€]						
MPC: N_H (time)		Building1	Building2	Building3	Building4	Building5	SUM	MEMAP
1	(15min)	51.9	27.6	347.2	220.9	182.8	830.6	664.4
4	(1h)	51.9	27.5	342.9	221.1	181.9	825.3	664.0
12	(3h)	51.9	26.3	340.4	221.0	180.0	819.7	664.0
24	(6h)	51.9	24.8	337.9	220.8	180.0	815.4	663.8
36	(9h)	51.9	24.4	338.5	221.0	180.1	815.9	663.5
48	(12h)	51.9	24.7	338.4	221.0	180.4	816.4	663.6

Table 6.1: The cost comparison for a 7-days period with a fixed electricity tariff.

The cost comparison (Table 6.1) shows three very interesting effects: The influence of the MPC horizon to individual buildings, the effect of MEMAP's coordination compared to individual buildings and the effect of the MPC horizon to MEMAP.

The *first effect* (the expected one) is that the optimized schedule of single buildings leads to less costs with an increasing MPC horizon, but only if flexibilities such as storage that shifts allows a better power balancing is available. This is the case, where solar power components are combined with storage. The effects of the cost oriented MPC is intuitive. Building #2 uses its battery to store energy, but only if it has the possibility to save costs within the upcoming MPC horizon. It means that the battery is not utilized, if costs are not saved. This is the case, when the demand is constantly lower than the production. Then the building sells the overproduction, since the storage has an efficiency that is less than 100%. The same is true, when the demand is constantly higher than the production in the upcoming MPC horizon. Then the optimal behaviour is to use the total production to cover the demand and to discharge the battery. Hence, charging becomes only useful when the two states interchanges. For instance, when the production is higher than the demand initially, but at some time point becomes lower within the visible MPC

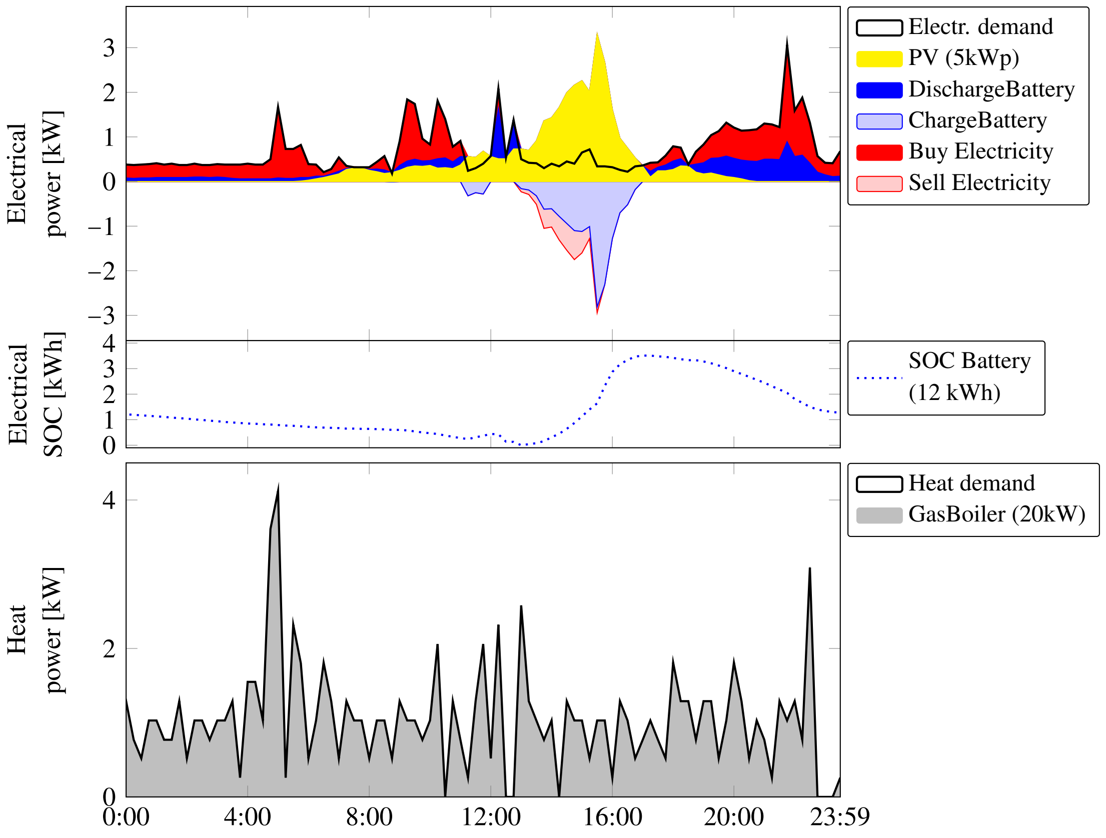


Figure 6.16: Operation of Building #2 in the individual mode with 6 hours MPC horizon (N_H=24).

horizon. Then the building stores energy to reduce the necessity of buying expensive energy. We see that this effect twice in Figure 6.16, once around 11:30 and later starting around 13:00. Due to the MPC horizon of six hours, the building plans with the upcoming demand until 19:00 and start charging. However, it charges initially not at full speed. It tries only to cover the visible residual demand during the upcoming MPC horizon. The six hours time horizon also prevents to plan the complete night time, thus, it does not use the battery at its full potential from the beginning. As a consequence, the planing of the charging becomes more efficient with longer horizons.

The *second effect* shows that interconnection of multiple buildings is beneficial for the operation. When all houses are coordinated through MEMAP, the total costs is reduced by about 20% compared to the sum of individual houses. The reason for the saving is that less efficient components are substituted by the more efficient ones (see Figure 6.17). For instance, the oil boiler of building #1 is replaced (i.e. it is always switched off) by the more cost efficient gas boilers. The gas boilers are again replaced by the more efficient CHPs. Further, the most efficient CHP (i.e. CHP of building #4) is used more often that the CHP of building #5. This replacement strategy leads to the strong cost reduction of 20%. It is repeatable for different settings and environmental conditions (see also Appendix B for further scenarios). Depending on the composition of components and electricity tariffs, we always observe a cost saving between 15% and 40%. But of course, this saving depends on the demands, available components and environmental factors such as solar radiation profiles.

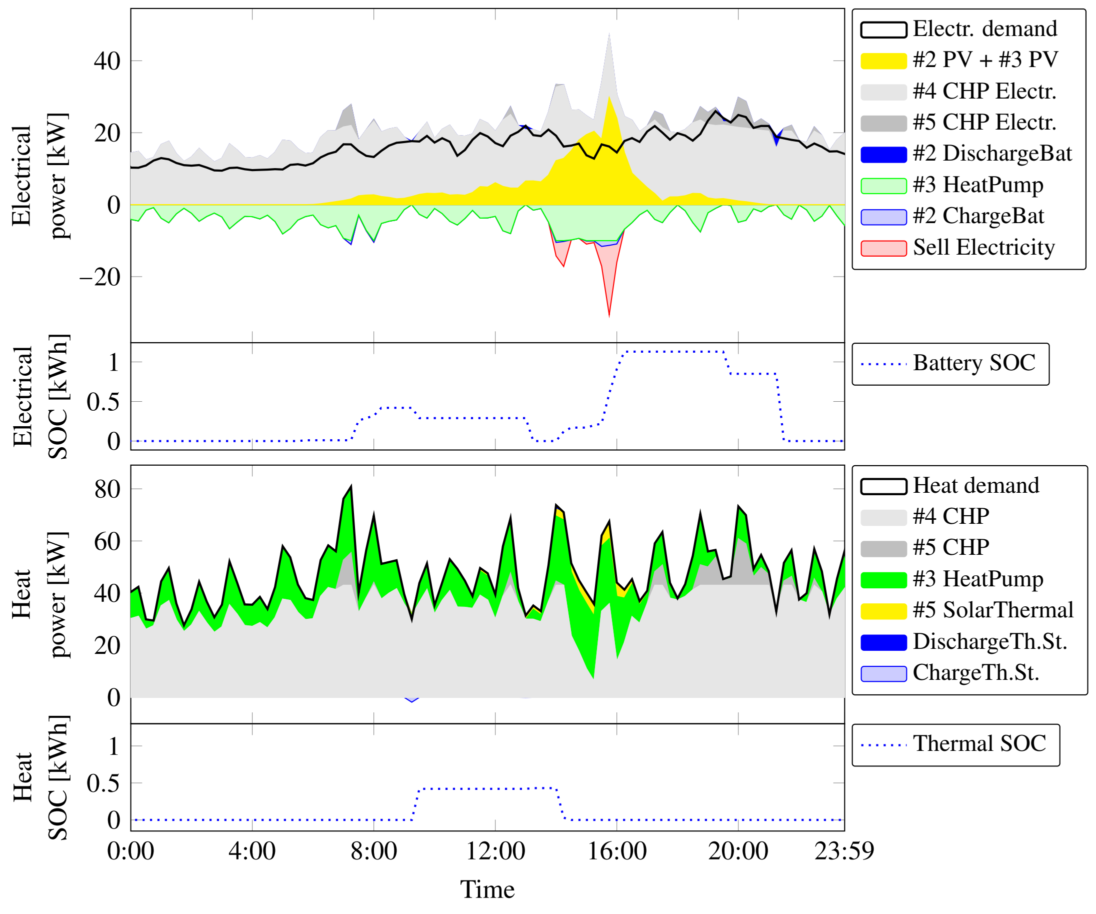


Figure 6.17: Operation of the quarter coordinated by MEMAP with 6 hours MPC horizon (N_H=24).

The *third effect* is, however, surprising. We observe that MEMAP's costs are almost constant for increasing MPC horizon lengths. The reason for that is that storages are not used that much in MEMAP, as the flexibility for time shifting the demand or production is encountered by a good distribution of power. In particular, MEMAP's MPC finds a very efficient combination of two components, the heat pump and the CHP. Those two components are a great combination and can cover most of the demand profiles[24]. The combination is so efficient, that it establishes almost a self-supply of heat and electrical power within the quarter. Additionally, it does not use the storage capabilities during the majority of the time (because of losses due to the charging efficiencies). As a consequence, the increasing MPC horizon length does not improve the costs. This is visible in the MEMAP's behaviour, which is presented in Figure 6.17. Note, that the thermal storage of all three multiple family buildings have a combined capacity of 300 kWh, and the battery of the single household building #2 has a capacity of 12kWh.

Further, it is also quite impressive that the coordination of MEMAP provides a complete coverage of consumption without the necessity to buy electricity from the grid. CHPs always produce heat and electricity, while the heat pump uses this electricity to complementary produce

[24]Note, that is particularly possible due to our assumption A7. For further remarks, see the limitations section of this case study further below.

additional heat very efficiently. Controllable generation components (not the couplers) are mostly neglected. To study this observation, we simulated further additional scenarios with different components, demand profiles and power tariffs. The combination of the CHP and the heat pump seems in fact very stable, throughout most of the scenarios:

- In scenarios with more renewable electricity production (e.g. large photovoltaic components, sunny days), the overproduction is completely sold to the grid, even if the price for selling is quite low. An increasing MPC horizon does not improve the costs.
- In scenarios with more renewable thermal production (e.g. large solar thermic components), the volatile heat production is used intensively and reduces costs, but surprisingly, the storage components are still not used that more intensively. The combination remains stable and the increasing MPC horizon does not lead to additional improvement.
- In scenarios with higher heat demand, the CHP and the heat pump combination is not changed as well, but the heat pump reaches its full operational limits more often.
- Only in scenarios with an increased electrical consumption (e.g. with electrical cars) additional power is bought from the market. In this case also the storage of electricity, leads to better costs with an increased MPC horizon. The reason is that buying of electricity can be avoided more often with batteries.

In summary, the *third effect* is that the combination of heat pumps with CHPs establish a very efficient operation, especially, when the heat demand is twice as much (in terms of kW) or more in comparison to electricity[25]. This ratio is today probably valid for most of the existing demand profiles. Only deviations from this ratio, thus, increased electricity demand, are covered with batteries to prevent the buying of electrical power.

The effect of variable prices for MEMAP

As a second evaluation we looked at scenarios with time dependent electricity tariffs. They allow more active prosumer participation, since storing, selling and buying of power at different times becomes one additional option to optimize the costs. To show the effect, we have decided not to use a typical dynamical price from the EEX spot market, even if this was technologically very easy to do. The reason is that MEMAP's behaviour profiles becomes very complex and are not easy to understand due to the high number of involved components. Instead, we show a simple price profile, where the price is constantly 0.05 [$^{€}/_{kWh}$] in the hours from 0:00 to 11:00 and 0.30 [$^{€}/_{kWh}$] from 12:00 to 23:00 and interpolated in between. We also use a good summer day as environmental input, to vary the environment slightly and show the full potential of MEMAP. The remaining parameters remain as before. The cost table for the second evaluation is shown in Table 6.2 for different MPC horizon lengths. The operation of the quarter is shown in Figure 6.18, the individual building's operation is presented in the Appendix.

In this flexible price scenario, storage components are heavily used. Electricity trading becomes attractive and also the heat is stored more frequently to increase the efficiency. We see

[25]The exact ratio depends on the CHP efficiency factors η_Q and η_P and the heat pump's *COP*-factor.

	Cost [€]						
MPC: N_H (time)	Building1	Building2	Building3	Building4	Building5	SUM	MEMAP
1 (15min)	47.7	5.8	95.1	224.4	178.0	550.9	366.1
4 (1h)	47.7	1.6	90.6	224.2	175.6	539.6	361.6
12 (3h)	47.7	-5.1	84.4	223.0	169.3	519.3	355.8
24 (6h)	47.7	-8.1	79.5	218.7	158.7	496.5	356.6
36 (9h)	47.7	-7.8	74.9	211.9	149.4	476.1	357.4
48 (12h)	47.7	-8.1	71.5	206.9	147.2	465.3	357.5

Table 6.2: The cost comparison for a 7-days period with a flexible electricity tariff.

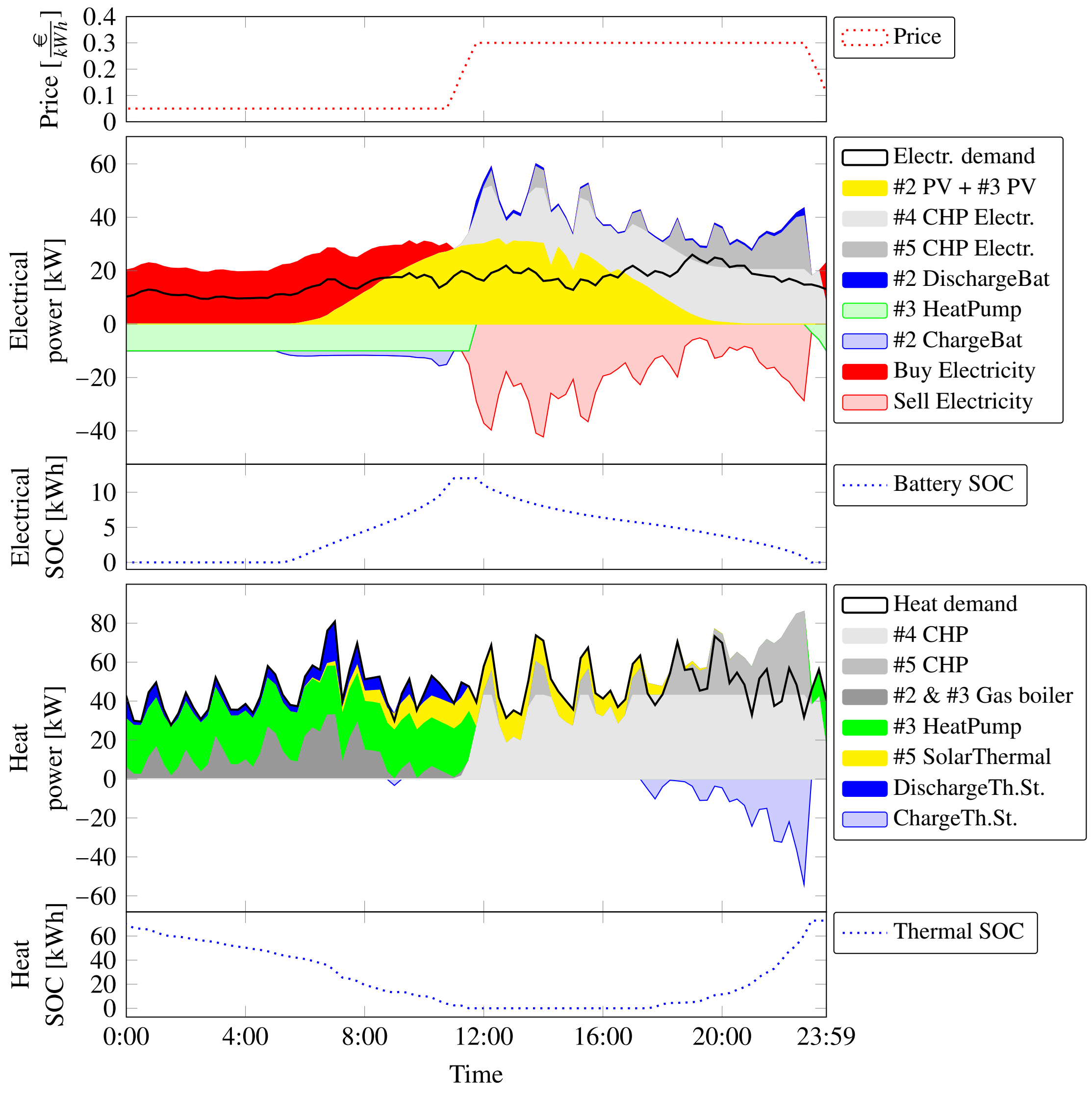


Figure 6.18: MEMAP's operation with 6 hours MPC horizon (N_H=24) and flexible price.

a number of effects. The first observation is that the MPC horizon matters, since it determines the operation of storages. The longer the horizon, the better storage components can be used. However, there is a limit. As soon as the horizon enables to utilize the storage components in their full range, its impact slows down. Further, we observe different preferences for the ranking of the components, that depend on the price level and the environmental conditions. Due to this ranking, the strong combination between the CHP and the heat pump decouples for the different price regions. Heat pumps becomes preferably used during low cost intervals together with gas boilers. The reason is that electricity becomes cheaper and the pure heat efficiency of gas boilers is better than those of the CHPs. CHPs, on the other hand, are preferably activated during high cost periods, since they additionally generate income through the production of electricity. Volatile energies simply increase the profitability, as the system takes their power without additional costs to increase its profit.

Even if the behaviour is different for the second scenario and we see a decoupling of the previous combination. Another major result remains as well. MEMAP achieves a cost saving of 20% to 35% compared to individual buildings, depending on the MPC horizon. Therefore, a coordination of several buildings, or in other words, the interconnection of individual EMS with a coordination system as MEMAP leads to better utilization of components in a neighbourhood.

6.6 Discussion

In this section, we discuss the findings of the presented case study. We reflect our case study steps to answer the stated research questions and discuss the threats to their validity. We also present the limitations of our approach to clarify the contribution.

6.6.1 Research Questions

In Section 6.2.2, following research questions are introduced as a motivation for this case study.

RQ4: When energy models are available, what are important details from those models to specify system interfaces and what are the possible transformation steps to develop an architecture of a coordination system?

RQ5: Which systemic effects can be expected from coordination systems?

RQ6: What do hierarchic coordination systems achieve in the context of SES and EMS and what do they not achieve?

We now discuss which parts of our work contribute to which research question and to which extend the stated questions are resolved.

RQ4: When energy models are available, what are important details from those models to specify system interfaces and what are the possible transformation steps to develop an architecture of a coordination system?

In Section 6.4.1, we introduce a short-term planning model similar to the EnergyHub approach presented Geidl and Andersson (2007). This model shows how operational planning of energy systems is carried out by power system engineers (see also Chapter 3 for more details). The presented model of Section 6.4.1 has two representations. The first representation is given by equations 6.2 - 6.5. Those equations represent physical constraints that determine the operation of the system. They are used to create the second representation of the model, which is the matrix form, i.e. equations required by the general optimization model described by 6.1. The matrix form contains the same details as the first representation, but we have to write them more explicitly. We did this with equations 6.6, 6.7, 6.10 - 6.17. This explicit matrix representation is not always presented in research papers (see also Bytschkow et al. (2019)). The reason is that the first representation is sufficient to describe the system and many solver can work with that already. It is shorter to write and its assumptions are easier to understand. Hence, for research this is the more convenient representation. However, to specify the communication interface, the matrix representation give additional insights. We explain why in the following.

In Section 6.4.2, we specify the interface of the coordination system that executes the planning process, i.e. solves the optimization equations 6.1 to create an MPC signal to the EMS. We use five different resource types for the specification. The reason for choosing those five types is found the matrix form, i.e. in equations 6.11 - 6.17. They very intuitively describe how each component is used and contain the necessary details for the specification. Further, the matrix form allows to understand the contribution of each resource and decompose the given information such that a very intuitive mapping of the model details to the interface can be derived. Hence, this representation gives more insights for the interface specification. Equations 6.2 - 6.5 can be seen as requirements to derive the matrix form. Their large benefit is that those equations can be easier analysed to understand the assumptions of system model and therefore the effects that are taken into account.

Further, we explain in Section 6.4.3 the data mappings of the input/output relation in a hierarchic structure. Hence, we explicitly show the system boundaries of the coordination system and related EMS from the software system perspective. This helps us to understand what a system boundary is. Such system boundaries are not available in the energy model perspective, i.e. Section 6.4.1. Instead the mathematical coordination model throws all the information into one place and assumes that this is the system of interest. It specifies what is inside of that model, but not how the system interacts with its environment. Therefore, the understanding of the term system boundary is quite different when we compare the work done by power system engineers and software system engineers (see also Section 3.2.1). That is one essential reason why many previous work on general SES architectures, such as the prosumer oriented architecture presented by Grijalva and Tariq (2011) or cellular power systems concepts presented by Benz et al. (2015) are that vague. They often miss to specify a concrete system boundary and therefore do not provide enough information about the envisioned system and an architecture to create this system. In order to do so, they need both aspects, the concrete model for the execution as in our Section 6.4.1 and the concrete interface specification as in our Sections 6.4.2 and 6.4.3.

To create the coordination system, we further constrain our architecture by the usage of well known architecture design patterns in Section 6.4.4. The architecture design pattern allow us to specify, how to integrate EMS into a group, which is realized by the MEMAP system. The MEMAP group remains flexible with respect to the number EMS components that belong to this group, but MEMAP has still a clear boundary, with respect to external systems. This allows to implement the ideas from Grijalva and Tariq (2011) or Benz et al. (2015) in real systems.

In our work we demand that an EMS is a server, with the specified interface towards the coordination system. This allows to access the necessary data from an EMS and to monitor its reaction on control signals, while the internal control system remains hidden. The choice of the protocol is not that crucial, but we suggest to use an industrial ready implementation. Thus, the second case study confirms that the proposed architecture design pattern are useful for the specification of the system and reliable for its implementation. This reinforces the generalization argument of our approach that was left open from the first case study in Chapter 5.

RQ5: Which systemic effects can be expected from coordination systems?

To analyse the effect of integrating EMS into coordination systems we have carried out a simulation in Section 6.4.6 using the scenario described in Section 6.4.5. The results are discussed in Section 6.5. Further results are shown Appendix B.

We described a situation, where advanced EMS that are capable to carry out their own optimized MPC are compared with an integrated approach, where a coordination system does the same, but in an integrated quarter scenario. The result is that the coordination system outperforms the individual EMS. In our scenario the improvement is in the order of 15%-30%. The exact number depends on time horizon, the environmental conditions and external prices. However, the integrated solution finds always a better solution as individual systems.

When we have a deeper look into the scenario execution, in particular in the time series results for the activated components we found that the reason for the better performance is mainly because MEMAP replaced the bad components first. In integrated quarters modernization of equipment within a single building becomes beneficial for the whole quarter. Thus, integrated solutions will be a reasonable approach that replaces bad equipment. The replacement will be quicker, if we assume the modernization activities within a group are more frequent as modernization single systems. The particularly nice side effect is that the increase of efficiency is beneficial for all participants of the quarter, the providers of the efficient equipment and those ones whose equipment is replaced. Further, integrated solutions open the path towards the usage of additional energy sources that are today hardly used, for instance the waste heat that sometimes occur in larger systems. This improves the energy utilization even further. Note, that the presented systemic effects are based on the introduced model. The model is only an approximation of the real system. We present further below how this threatens our statements for that research question.

RQ6: What do hierarchic coordination systems achieve in the context of SES and EMS, and what do they not achieve?

The last research question is a retrospect on the general idea of having a hierarchical system to coordinate EMS. In Section 6.3 we discuss the hierarchical structure of this case study and work upon that structure throughout the different steps in Sections 6.4.2 - 6.4.4. We see that a hierarchy is beneficial to establish the desired coordination in different ways.

One benefit is that we capture the right system boundaries and are able to describe system interfaces between the different hierarchical layers. This provides a clear understanding about the functionality of each hierarchical layer. In our case study the coordination system is responsible to improve the operational control of the quarter. The individual EMS are responsible to control their devices. The data exchanged between those two layers is given by the interface specification. This separation enables to apply two fundamental principles that are important for software engineering, the separation of concerns and abstraction. We particularly emphasize this point in Section 6.4.4.

Another benefit of the hierarchy is presented in Section 6.4.5. It describes a group that is created. Section 5.4.3 shows another similar example. We demonstrate that the notion of a group and a system coordinating that group is interrelated. The benefit of the hierarchy is that it supports to reason about groups, their system boundaries and their behaviour. This means that we can clearly distinguish which behaviour belongs to the inner behaviour of the group and which behaviour belong to the externally visible behaviour. We work on that also in Section 6.4.3.

Nevertheless a hierarchical system is not a one size fits it all solution. Section 6.4.5 presents our assumptions to develop a coordination for the group of buildings. In that section we assume that power can be freely exchanged between the participants (A1, A7). This limits the hierarchic system as follows. As long as we consider only one area, coordination is able to take place as presented. But, if we have two areas and each one has it own district heating, coordination becomes more complex. The coordination of two distinct areas is possible, but it requires to introduce a network topology that limits the power exchange. We have presented such an extension in (Heidemann et al., 2019). In this extension we can model that both groups share one electrical grid, but have two distinct heat networks. The MPC considers the corresponding constraints. However, the information of the network topology needs to be available to the coordinator, but it is not clear whether this information can be provided by the children of the coordinator or not. Therefore, the hierarchy cannot be simply extended to the higher levels, without taking such considerations into account.

Therefore, as answer to the research question, our approach and the hierarchical structure helps due to presented reasons. But it is not necessarily the one fits it all solution. We have to be sure, that the system that is being controlled meets the physical requirements, otherwise the hierarchic coordination is not possible.

6.6.2 Threats to Validity

In the following, we present the threats to our case study results. We use again the three categories construct validity, internal validity and external validity as introduced in the previous chapter.

Construct validity: To answer the research questions of our case study, we created a coordination system for a quarter. For the creation we used a classical multi-energy coordination model in Section 6.4.1. It was used to derive a coordination system with a hierarchic architecture.

One threat is that we have not selected a representative model. Instead we used a model that is not suited for coordination. Therefore, it should no be used to specify the interfaces of the EMS. We tried to mitigate this threat by using a model that is used in a larger research project setting, where different research partners, among them power system engineers and industrial partners with experience in district heating and EMS providers discuss the model intensively and work on extending this model together. We also presented the model at two different conferences (Bytschkow et al., 2019; Heidemann et al., 2019), where we got positive feedback on our approach. One could also argue that another model, for instance for distributed control is better suited for coordination. This might be indeed the case. Of course in distributed control, the data at the interfaces will change. Nevertheless, our main argument to use models for the specification of coordination behaviour and the interfaces remains also valid for that case.

A second threat is that our approach that is based on several steps is not suited to create a system architecture for a system that integrates EMS and coordinates them. Hence, our work describes a non realizable system. We mitigate this threat by creating a real coordination system that integrates several EMS with real devices. Among them, there is a large experimental laboratory COSES at the Technical University of Munich[26]. It has real devices and real networks for extensive studies of control models and control systems. Actually, our integration effort aims at creating a reliable environment to study the validity of coordination models, including the presented model but also its future versions in real experiments and to reason about the system architecture for coordination system. The question remains, if our solution by specifying the interfaces based on the matrix form of the model is easier for other partners as well, since so far many transformation steps from the model up to the system architecture were done by us. Therefore, we can only argue for the ease of transformation from our perspective.

Internal validity: Our case study provides quantitative effects that are reached by our system. These claims are of course only valid for the presented model and the selected scenario. We present both in Section 6.4.5. Several threats remain open.

The accuracy of the components for our coordination model are quite abstract. Our model is linear, meaning that a component might take any states between zero and its maximum limit. The threat is that not all components are like that. Especially CHPs, oil and gas boilers should be able to be turned off or operate in a defined bandwidth. This can be solved by choosing another optimization model for the MPC, for instance a MILP. This leads to different results. However, we saw in our time series analysis that components, like gas and oil boilers and also storages are used less frequent as in single systems. In combined systems like the quarter, the efficient components dominate and they run at higher operation set-points. This effect favours our claim for using coordinated systems even more. Another accuracy issue of the model is that we assume a fixed efficiency. Also this assumption can be removed, if we create a non-linear optimization model. The non-linear model, thus, would give more precise results for Section 6.5. But our architecture would not be affected by that. Further, in current district networks, for instance from the local utility, the temperature level for the heat supply line is around 80-90°C (SWM, 2015). Not all components can feed into that network. As a consequence, MEMAP should only take those resources into account that can technically feed into the network. However, future district heat networks are expected to operate at much lower heat temperatures, for instance at 45°C (Lund et al., 2014; Wesche et al., 2017). We therefore expect, that most of the components

[26]Center for Combined Smart Energy Systems (COSES)

might be able to inject heat power. Further model accuracy issues can be considered. For instance, losses for electricity and heat can be considered. Dynamics can be added to components in order to tweak the model towards reality. However, those tweaks are more a parametrization issue of the model, not a conceptual problem of our approach. Also, we intend to work with partners to use MEMAP in an experimental environment, where we can validate the model and our assumptions in real experiments, as presented before.

The second possible threat for the investigated coordination is based on the available data set that we used for the parametrization of the scenario and as demand profiles. This might result in wrongly chosen devices and unrealistic consumption and production profiles. We discussed this issue with our project partners several times. To mitigate this threat, we went with the project into a field test in Riemerling, which is a suburban area close to Munich. This area should be the reference for our case study. We analysed five buildings together with our industrial partners to create a better scenario. We observed indeed that many generation units are oversized. Unfortunately, the generation units are rather old and similar to oil and gas boilers. Modern devices like heat pumps or CHPs were not present. This is however, important to show the potential of coordination particularly for a multi-energy scenario. To address the issue of better data we prepared a native connection of MEMAP to the meters of the buildings. For that we use our system with the described interfaces (see Section 6.4.4). We analysed the available interfaces and our architecture allows this interconnection. However, at the time of writing, the data from the field test was not available, therefore, we relied on the generated data provided by our research partners (i.e. Jambagi et al. (2015); Kramer et al. (2016, 2017)). Nevertheless, from the technical point of view we are able to integrate also real field tests into our system. Therefore, available data might increase the confidence of the results, but it not a conceptual issue of our system architecture.

External validity: The argument for generalization is supported by this and the previous case study. The main threat here is that the number sample sizes is too small and too synthetic due to a limited number of real EMS. To mitigate this thread we worked with industrial engineers and other research groups from different domains. We integrated several EMS, from research and from the industry. The integration was not always possible with OPC UA, since other REST API were in the field as well. However, the general representation including our architecture design pattern was possible. When scanning through the different EMS, we got the impression that all systems are indeed quite similar and that with time we will have to possibility to coordinate them.

6.6.3 Limitations

Finally, in this section we discuss the limitations of our case study. Some of the limitations can be eliminated by extending our approach with additional functionality, that were not in the focus in the presented case study. Other limitations exist due to practical reasons and the current technical state of the art.

EMS schedule commitment: In our case study we introduce MEMAP. This is a system for coordinating EMS. Its intention is to collect the data from the individual building EMS and propose a better, coordinated scheduling. If the building follows this schedule, we obtain the desired results (see Section 6.5). We only show the case where each EMS behaves as expected. We do not show what can be done when a building does not follow the signal. This however is important in real systems, since without considering such cases, the coordination will not work. A solution might be to consider only those buildings that follow the signals provided coordination system, since its participation in MEMAP can also be seen as a commitment to provide resources. Buildings that ignore the signal can be temporarily excluded and do not profit from the savings.

Deviations from schedule: In this case study we present an MPC that considers planning with some time horizon (see Section 6.4.1). Even though the MPC can be updated regularly, for instance once every minute, it has not be designed to close potential deviations from the schedule. Its task is only to plan the supply and expected demand accordingly. Therefore, a parallel control application to close the deviations is beneficial. We have presented a possible solution in the first case study (see Section 5.4.2). Hence, a combination of both case studies would be desired, that considers potential side effects (e.g. SOC) as well.

Internal EMS behaviour: We understand an EMS as a black box component (see Section 6.4.2 - 6.4.4). It provides its resources at the interface, but we have no information about its internal system details. This is an intentional design of our system and we think that also in future, coordination systems will have to rely only on the EMS data that is provided over the interface. Of course, to understand the technical implications we need also to study how the EMS behaves internally. Additional essential technical design decisions are required to create a system with the technical possibility for power injections into the network. Particularly, also the consideration of the district heating temperature limits our approach, as some components might not be able to inject their power that easily. We discussed this issue a lot in the project with our industrial and power system engineering partners. One possibility is to filter out such components from the interface of the EMS or from the MPC, hence neglect them in the coordination. Another possibility is apply MEMAP in quarters with low temperature district heating. But these questions go beyond the scope of a software architecture. We leave them for future work.

Business cases: Our case study demonstrates that in an integrated solution operational costs are reduced (see Section 6.5). This immediately opens the question how those savings are shared between the building operators and how the coordination system is reimbursed. It also opens several questions with respects to a return on investments along modernization activities. In our case study we do not discuss those points and leave this as its own topic.

Interoperability limitations: The final limitation is important for the system developer. Our case study assumes that each building has an EMS with an OPC UA server. But we will usually not find this situation in real systems yet. To reach the interoperability with MEMAP easy modifications are possible. A possible technical extension is to use the same interface specification, but implement it using other available APIs, e.g. with REST. Since REST already exists in many solutions the modification is quite easy to achieve. Often, REST based interfaces use the JSON serialization as the main data format, exactly as we do in our internal messages within MEMAP. This allows to convert REST interfaces into the desired message format. Actually, this has been used in Section 5.4. Therefore, this limitation can be hurdled by adapting the protocols and parsing the data according the desired coordination model. It is not a conceptual problem.

6.7 Summary

We presented a case study in the area of sector coupling to create a hierarchical coordination system using the approach and architectural constraints of the preceding chapters. The case study demonstrates how a rather complex scenario with a high variety of different energy components is controlled using our generic architecture. The case study provides a research contribution in two important areas.

The first contribution is a description of a stepwise approach, that uses mathematical models for coordination to specify coordination systems, particularly the system boundaries that matters, namely the interfaces of those systems that are being integrated and their behaviour. In contrast to other approaches, which focus either on the behaviour (e.g. MATLAB optimizations, development of controllers) or interfaces (e.g. standardization activities), we demonstrate how the two important areas can be addressed in a consistent manner. Such discussions are of paramount importance to clarify the boundaries of the involved system and to provide concrete information for system developers that implement real running systems.

We further show the implementation of the presented solution with a prototype that exists in two stages. The initial stage is a simulation. It provides a framework to simulate and evaluate different scenarios and to determine the potential for sector coupling and interconnection of quarters. The simulation uses a linear model to calculate an optimized schedule. The MPC with the linear model is just an example. If the accuracy is not high enough, the model can be exchanged with other, more precise optimization approaches, for instance mixed integer programming or other non-linear optimization methods. In that case, we might need to refine the interface, thus, extend the available data points. This, however, is straight forward. The improved stage is a sever based platform. It provides real industrial interfaces and a realistic environment, while the internal logic is the same as in the simulation. The platform can be used to study hardware in the loop system or even control real EMS later on.

The second contribution is the demonstration that we achieve a better efficiency in combined quarters. To make such statements, we compare the best possible operation of individual buildings with the combined quarter solution. We see cost savings in the order of 20% and more depending on the composition of the devices and the environmental conditions. The improvement is mainly achieved by substituting some of the worse components with better ones. We also observe promising combinations of components for different environmental conditions. The case study confirms, that a higher energy efficiency can be achieved when quarters are coupled together. Our approach is easily extensible to other scenarios and other device combinations so that we are able to use our system for further studies of smart energy system solutions.

Finally, we would like emphasize that any system description without a clear system boundary specification remains vague even if the general idea of combining different systems seems intuitive at the beginning. A way to reduce this ambiguousness is to define clear system interfaces and a clear behaviour. This is the only way to create systems as envisioned by Moslehi and Kumar (2010); Grijalva and Tariq (2011); Benz et al. (2015) and achieve a higher contribution of smaller prosumer systems with additional flexibility.

7 | Conclusions and Outlook

This chapter summarizes the findings and contributions of this work and provides an outlook for possible future research directions.

7.1 Conclusions

The aim of this thesis is to improve the capabilities to integrate energy system components in coordination approaches and by that increase control capabilities in power systems. The increase of control capabilities is of paramount importance for environmental friendly and reliable power supply systems with many renewable energy sources. The reasons for that are explained in the motivation of our work. The current state of the art has a number of open conceptual and practical problems that hinder the integration:

- *Problem 1: Integration with current technologies,*
  particularly in context of secured SCADA systems that need to be opened and extended with the possibility to integrate more consumer oriented devices.

- *Problem 2: Ambiguous architecture specifications,*
  particularly when similarities of components are the only description (e.g. Moslehi and Kumar (2010); Grijalva and Tariq (2011); Benz et al. (2015)), but other architectural questions like system interfaces and system behaviour are neglected.

- *Problem 3: Specification of EMS system boundaries, interfaces and coordination models*,
  particularly because of the different understanding of system boundaries of power system models compared to software system models.

- *Problem 4: Evaluation of hierarchic architectures for EMS coordination*,
  particularly because of the missing technological support to evaluate such architectures and that current coordination models focus on devices rather than on EMS with an interface that represents the available control flexibility.

To approach these problems, we introduce a number of contributions. First we analyse current power system coordination models that are of interest for coordination. We then provide an approach to specify a generic hierarchic architecture to coordinate EMS, i.e. to specify the interfaces and the behaviour of EMS that are part in a hierarchic structure of interacting components. We use well-known architecture design patterns to explain and support the creation of such systems. We further complement the approach with a technological framework to study such architectures in a follow up step. We validate our approach with two case studies that integrate systems with interfaces that are also used for the integration of industrial systems. The case studies answer several research questions related to hierarchical coordination systems in the energy domain.

Analysis of SES models and their usage in hierarchical structures

Mathematical models are the essential basis for power system planning and operation. To create an architecture we presented and described classical energy system models that are of interest for coordination. We particularly focused on the data that is available within those models. In the next step, we worked out the notion of a system boundary of energy system models and its different meaning compared to the system boundary of a coordination system, which is always a software-based system with a clearly defined input and output interface. We provided several examples for interfaces that are specified such that an energy model can be derived from the data available at the interfaces. Finally, we worked out how the specification of such an interface is transferred into a hierarchy of coordination systems. Since system boundaries, i.e. system interfaces, are important to specify architectures, this analysis provides several missing details, i.e. the necessary data at the interfaces, to approach the problems from above.

A generic hierarchic architecture for the coordination of EMS

To develop an architecture that is used for the coordination of EMS in real systems, we provide a detailed step-by-step derivation of architecture constraints that are required to build such a system. The derivation includes the following steps:

- Identification of important requirements containing the current technological state of practise for energy systems and required extensions for SES.
- A stepwise consideration of five architecture design patterns to address the previously identified requirements. The architecture design patterns include the *client-server* model, *composite pattern*, *layered systems*, *unified interface* and the optional (but beneficial) *publish-subscribe* pattern.
- A behaviour description using a distinction of cases for hierarchical systems, where parent components interact with child components. The distinction allows to describe the behaviour of EMS in a hierarchy based on atomic behaviour cases. It therefore helps to clarify the possible interactions in a hierarchy and simplifies the specification of coordination systems with behaviour specifications based on input to output mappings.

The presented architecture design patterns and the case distinctions as interaction specifications between software systems are the basic constraints to define hierarchic architectures for the coordination of EMS. They provide a clear specification and additionally help to discuss architecture design decisions at an early stage of system development. This reduces ambiguity that we see in other specification activities and eases the system implementation later on. Further, the selection of well-known architecture design patterns allow to choose certain technologies, i.e. available libraries or protocols, for the implementations. This speeds up the development process and reduces errors in the implementations, because we rely on well tested and sound technology stacks from other domains that are used in many industrial applications.

A framework to implement and test hierarchic SES architectures

To facilitate the practical usage of our approach, we developed a framework to implement and test concrete scenarios. The framework is designed as a co-simulation framework. The framework is actor based, i.e. the behaviour of software systems (e.g. an EMS) is represented as an actor. Actors do not share their internal states and interact only via messages. They are therefore a representation of distributed systems as we see in reality. The framework supports:

- Implementation of EMS that are arranged in a hierarchy and interact only via messages. The messages can be automatically converted into interfaces, e.g. REST or OPC UA.
- Consideration of power system topologies using a standard representation with CIM.
- Solving power system equations with external simulation tools, such as Gridlab-D, but also with internal solvers.
- Interaction with external systems using the *client-server* model, where each actor has the possibility to have its own server or its own client to provide data for external systems or to obtain data from other external servers, respectively.

The framework is used as a technological vehicle to implement the presented simulation scenarios. Beside the presented work, the framework was used to study Distributed Ledger Technologies (DLT), also knownas blockchains, by Thut (2018); Lumani (2018); Bajpai and Duchon (2019), where a set of EMS (actors) interacted over DLT to offer bids for supply and demand. Another usage of the framework was to exchange observations for machine learning scenarios to implement and evaluate whether reinforcement learning is suited to learn the coordination by Andres (2019). The framework is therefore suited to study different concepts for the SES domain.

Case study: Hierarchical architectures for virtual power plants

To evaluate our approach (specification of a hierarchic system of EMS), we worked on two case studies. The first case study is a hierarchical VPP that consists of clusters of components. Clusters are used to create a representative for a group of components. The group is used to keep a schedule for power production on the one hand and communicate the remaining flexibility to the higher hierarchy level on the other hand. This case study shows how a hierarchic architecture is created. The interfaces are designed to offer control flexibilities. The behaviour is a coordination algorithm that is responsible to activate flexibilities in order to keep a desired schedule. The case study answers the following research questions:

RQ1: How does the hierarchical concept affect the integration of interactive EMS and the architecture of composed energy coordination systems? We demonstrated that the most important aspect for integration is the definition of the EMS interface. The interface defines the system boundary. For the hierarchy, it is necessary that the interface specification applies to multiple hierarchical levels. Thus, the interface should be specific enough so that leaf EMS are able to represent their devices, but at the same time abstract enough, so that group representatives (cluster systems) use the same interface also towards their parents. To specify the system's behaviour, the input to output data processing should be clear, particularly for intermediate cluster systems. Our behaviour cases structure the specification activities.

RQ2: Does the hierarchy support different energy coordination mechanisms? Yes, we presented three different coordination mechanisms to demonstrate such support. The shown coordination mechanisms vary in their complexity and in their accuracy (speed of establishing the desired system state). Nevertheless, they integrate the systems using the same interfaces. This confirms that interfaces are essential to create a feasible specification.

RQ3: How do the proposed architecture design patterns help us with the development of such coordination systems? We evaluated whether the proposed architecture design patterns fit to create a feasible technical solution. We found that the architecture design patterns do help with that. They help to understand the system's composition and the interaction between the composed systems. Further, they allow to select suitable technologies and implement a running system. They also support the generalisation of our approach.

Case study: Architectures of coordination systems for quarters (multi-energy systems)

The second case study applies our approach in the context of multi-energy systems (i.e. sector coupling). We focus on the integration and coordination of more complex EMS. The goal is to improve the energy efficiency in a quarter based on available supply and demand that exist in participating buildings with the specific requirement to combine different energy sectors (heat and electricity). We present a technical solution and answer the following research questions:

RQ4: When energy models are available, what are important details from those models to specify system interfaces and what are the possible transformation steps to develop an architecture of a coordination system? Energy system models for coordination describe the matching of available energy supply with the expected demand. The models are expressed as a set of equations. Because of classical solving methods (e.g. LP or MILP) also matrix representations exist. These representations give us additional information for the input / output data of the optimization. The behaviour specification goes along that. We have shown several transformation steps how to derive the hierarchical system from an short-term planning model including an automatic generation of OPC UA interfaces.

RQ5: Which systemic effects can be expected from coordination systems? To derive quantitative numbers, we compared two operational scenarios, one where EMS of single buildings optimize their own behaviour, and a second one, where the coordination system does it for a larger setting. We could confirm that an integrated always system outperforms single buildings, if it has the physical capabilities to exchange the power. The main reason for that are additional options to select efficient equipment for the operation. The improvement depends on the devices and the environmental conditions. In our scenario we observed savings in the order of 20% and higher.

RQ6: What do hierarchic coordination systems achieve in the context of SES and EMS and what do they not achieve? Our architecture has several benefits. It identifies the system boundaries of relevant systems quite successfully. This allows to specify interfaces between the hierarchical levels. It also helps with the separation of concerns and abstraction. Further, it allows to establish coordination groups. The groups support each other, when this is physically possible (i.e. transmission lines exist and nodes inject power bi-directionally), if not, the hierarchy does not offer additional benefits.

Further findings

Our work demonstrates that hierarchical architectures are technically feasible for energy systems. Our two case studies explore control and planning coordination applications for energy systems. For control, we show that flexibilities are useful to cover deviations. We specify the interfaces and show coordination mechanisms for that. For planning, we show an extended sector-coupling scenario with multiple buildings. Therefore, our conclusion is that hierarchic architectures are reasonable.

However, in contrast to the previous work, where the authors assume that a generic hierarchic architecture is simple solution for the energy domain (e.g. Moslehi and Kumar (2010); Grijalva and Tariq (2011); Benz et al. (2015); Reuter and Breker (2018)) we see that concrete specifications are still complex. Therefore, the expectations for simple solutions are probably exaggerated. The complexity is hidden in the necessary level for details in this technical domain. For instance, a flexibilities from the first case study is modelled as activation profiles with a fixed price. This is only an approximation. Many more details can be added to the specification. For instance, flexibilities can add a non-fixed price, e.g. due to a non-linear efficiencies of the components' production. Such details improve the model accuracy, but increase the complexity for the aggregation and decomposition of control signals for the lower hierarchical level. This diminishes the benefits of a simple hierarchy. Similar effects are present in the second case study. For instance, thermal components like heat pumps can only inject power into networks with suitably temperatures. If the temperature is too high, heat pumps have to be omitted in the operation. Of course, this can be considered in a rigorous algorithm. However, such details require more expressive EMS systems and the underlying infrastructure needs to support these kind of operation schemes, i.e. with higher and lower supply temperatures. This requires further analysis in the future from power system engineers and software system engineers. Nevertheless, we see that coordination provides better results than individual buildings, therefore, integration and coordination of EMS is an important technology to improve our energy systems.

7.2 Outlook

In this section, we outline possibilities for future research activities that improve or extend the presented contributions.

Application of our approach in extended SES scenarios

We present two case studies. Each case study evaluates one particular coordination type, i.e. control or planning, but not both at the same time. The combination of control and planning in one system would provide further insights. It would define additional data points for the design of EMS interfaces. The implementation of an extended solution is therefore quite interesting. Further, we observed further hierarchic applications with self-similar interfaces in related projects. For instance in the work by Gupta and Duchon (2018) the same structure is used to define applications for restoration scenarios in power systems. Such applications provide additional insight in the application of hierarchical systems and extend them to further scenarios. We are also quite confident that our work is beneficial for the specifications of the cellular energy system, which is currently discussed by the VDE (German Electrical Engineering Association). The reason is that the presented concepts are still quite close to the modelling approach (as presented by (Benz et al., 2015)). Therefore, it would be quite helpful apply our approach in such projects.

Evaluation and validation of the models in real field tests

The current specification of our interfaces is deduced from an MPC approach, which is a classical way to show the potential for an improved operation of energy systems. However, the models that we use are of course only approximations. Therefore, it is desirable to validate the accuracy of the models in real experiments. This allows to determine and validate which factors are additionally required for the model and which are not that important. Actually, we plan to validate our second case study in a real laboratory with our partners. This laboratory is currently created at the Technical University in Munich at the CoSES Research Center[1]. In this experimental environment, we will be able to investigate the second case study in more detail. But, this is future work for multiple researchers from the power system and computer science domain.

Using AI for EMS coordination

Our coordination relies on models that contain significant modelling knowledge of power system engineers and depend a lot on specific scenario details. When modelling such a system, it is always a lively discussion, which details are necessary, which parameters are considered in which accuracy and which ones can be neglected in our coordination approach. Hence, the modelling itself is quite time intensive and the details impact the required computational power to calculate the optimal schedule for the components. The available framework allows to obtains real data and apply machine learning techniques. The work of Andres (2019) shows the first steps in that direction. It is of high interest to study if this can also be applied in a real environment. This would reduce the modelling effort, and in addition, it could be a generic way to capture physical models and topologies for coordination systems.

Transfer of the approach to other domains

Finally, our case studies evaluate the hierarchic architectures for coordination specifically for the energy domain. Although the results indicate that the approach if applicable in this specific domain, further evaluation is required, if we want transfer our approach to other domains. Especially those domains are of interest that rely on the planning of resources and can be structured in different hierarchies. Possible domains could be logistics, mobility, city planning or even financial products. We expect that these domains require adapted models and new specifications for the interfaces. Nevertheless, our approach might provide new thought-provoking impulses, particularly for planning tools and systems for monitoring and control in these domains as well.

[1]CoSES: Center for **Co**mbined **S**mart **E**nergy **S**ystems

Appendix

A | Additional hierarchic architectures for prosumers

Complementary to this thesis, we developed within our research group an EMS for a prosumer to control the appliances that are part of our research environment. The developed system is a layered, modular, component based automation system with a focus on energy efficient operation. It hierarchically integrates an extensive set of heterogeneous devices and runs in a distributed environment. We call the system SMG 2.0 [1]. It received several iterations from (Koß et al., 2012), a SOA based prosumer system, to (Duchon et al., 2014) an improved sensor aided SES node.

Our EMS controls an SES node that consumes, produces and stores energy. It is a classical prosumer system. Our hierarchical concept is a main architectural style to provide monitoring and control interfaces (APIs) for applications. We developed several EMS services, that integrate various sensors, actuators and different devices, which can be controlled to shift or adapt their energy consumption. Flexible devices, such as a battery, provide additional operational flexibility. We use them to optimize the building behaviour based on internal optimization criteria (e.g. user settings) or on external stimuli (e.g. flexible prices), depending on the set up of our experimental settings. This allows to react on external price signals, operate as a self-balancing system node or participate as a component in demand side management solutions.

Research focus

The general aim of the SMG 2.0 system is to provide an experimental environment and collect experience for the investigation of different smart energy problems and possible solutions. The aim can be broken down into required functionality including: (i) technical solutions to integrate our physical hardware (sensors and actuators) and their deployment on various system platforms; (ii) conceptual solutions like interconnection to different stakeholders, energy markets or neighbourhoods; (iii) algorithmic solutions like forecasts and optimized scheduling of devices; and (iv) solution for human machine interactions (Duchon et al., 2014). The evaluation of those solutions is part of the developers' research activities.

Experimental setting

The SMG 2.0 system was developed at the research institute fortiss. It is the core software system of the Smart Energy Living Lab to control and monitor various distributed devices. It provides a variety of functionality for an energy efficient operation, its automation as well as an interface for user interactions with the Living Lab. The Living Lab is a retrofitted office building with various sensors, actuators and equipment for energy production and storage. We distinguish between four categories of components that are available in the Living Lab. They are shown in Figure A.1.

The first category includes energy production and storage components. The Living Lab has photovoltaic solar modules with 5.4 kWp installed power on the roof. It is connected with an inverter to a number of selected office rooms of the building. In addition, we have a 8 kWh

[1] Smart Micro Grid 2.0

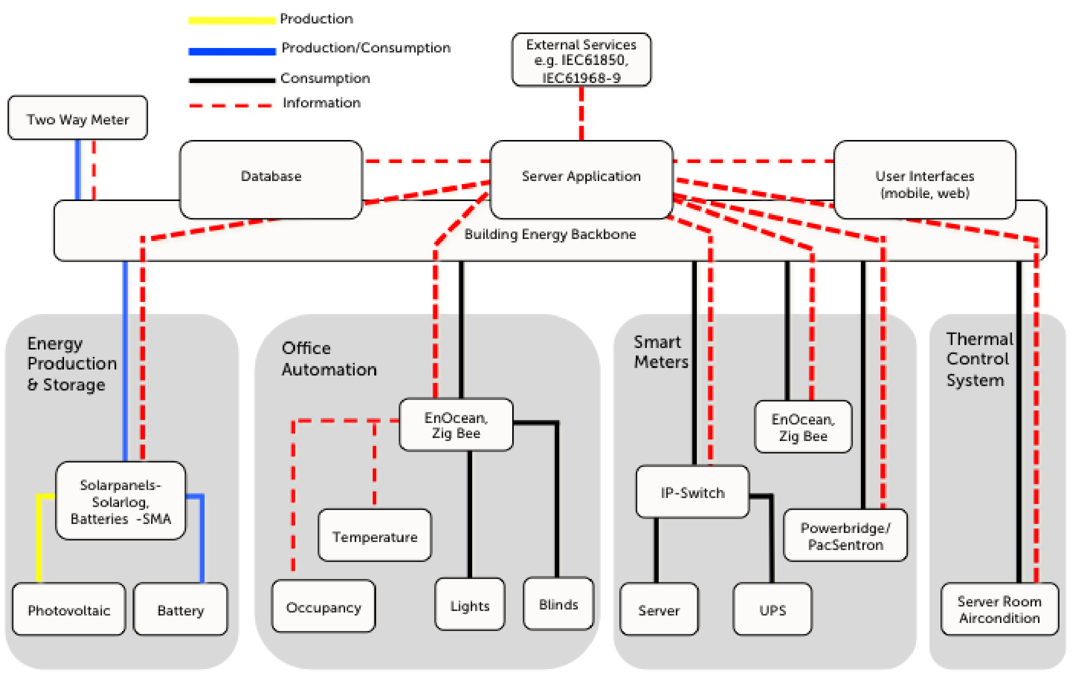


Figure A.1: Smart Energy Living Lab: Technical overview

LiFePO$_4$ battery system that supplies the same rooms. The Living Lab office space is extended with additional energy data logging equipment that we process with our SMG 2.0 system as well.

The second category subsumes all the devices that belong to the building automation domain. This includes the rich variety of sensors in the rooms, like temperature, humidity, occupancy, brightness, smoke detection, and actuators for lights, blinds or power sockets. The equipment from the second category covers a broad range of technological solutions, like KNX, Dali, BacNet, Modbus, EnOcean, ZigBee, Z-Wave, etc. Depending on the use case, we used wired or wireless solutions for building automation.

The third category of the Living Lab covers the field of smart meters. They provide relevant energy data and protocols for Energy Data Management (EDM) applications and are expected to act as a gateway, which aims to provide a rich functionality within the Home Area Network (HAN). We use various vendors with different technologies in the Living Lab to understand the implications of smart meters for the IT systems within the utility sector. Our equipment includes smart meters from Fröschl, PacSentron, IP-Switch, EnOcean and Hexabus plugs.

Finally, the fourth category of devices is dedicated to thermal components, including heating and air conditioning. This allows us to quantify the thermal demand of buildings and its users. We can also identify additional flexibility of thermal energy demands to support functionality for valley filling, or peak shaving by temporarily shifting the device's operation. Further applications in this category are related to the field of energy coupling, in which previously disconnected infrastructural domains like electric and thermal installations are interconnected. This allows further optimization of the energy efficiency, and additional strategies to improve the infrastructure reliability.

All devices of the four categories are controlled by our software system. For its implementation multiple requirements were important. We stated them in (Duchon et al., 2014), but in order to understand the rational for the chosen architecture and how the hierarchical architecture helps to address those requirements we give a short summary in the following.

Requirements

The core challenge was to design a system that can be developed stepwise and be maintained as easy as possible, as our research projects often requires modifications, integration of new components, and new functionality from the rich variety of energy related research tasks. We therefore decided to have a very modular design. Beside that, we had a number of additional requirements due to our living lab environment. The system should be: (i) able to handle various, distributed sensors and actuators, independent of any vendor, (ii) able to gradually integrate new functionality, (iii) able to build functionality that uses available functionality, (iv) extensible so that new components and devices can be added or removed in an easy way, preferably at runtime, (v) simple to configure, maintain, and troubleshoot, (vi) scalable, so that it can handle a lot of communication and data exchange, (vii) able that the user adjusts the trade off between energy efficiency and user comfort, (viii) offering various interfaces for outside communication, e.g. to exchange data with mobile devices, energy markets, aggregators or utilities for ancillary services.

Based on the requirements and a previously available demonstrator that was based on a layered, service oriented architecture (Koß et al., 2012), the system was adapted to enable more flexibility with respect to the interconnection of functional components. This step also introduced the hierarchic architecture as a generic principle of the EMS, which introduced an API for the design and interconnection of new components within the system. We explain the hierarchic concept as part of our architecture in the next step.

Generic architectures for prosumer EMS

The SMG 2.0 system is based on a modular, component based architecture that uses hierarchical structures with similar interfaces in multiple software components. The provided functionality is encapsulated in loosely coupled components. They can be developed and deployed independently on a single machine as well as in a distributed system environment. The interconnection of components is purely message-based. We use a scalable message brokering system that handles the distribution of the messages between the connected components. The binding is based on a publish-subscribe mechanism, that connects output messages to input queues of the connected components. All messages can be also sent without an established binding, but they won't be delivered. This loose coupling eases the development and deployment of components and enables to start and stop components independently of each other.

The available system components are shown in Figure A.2. We have chosen a layered architecture to reduce the complexity for the interaction between the components and abstract from the large heterogeneity of the hardware devices. We established four layers based on their functionality. The lowest layer is responsible for the integration of hardware devices. It provides an abstraction from the sensors and actuators and harmonizes their interaction with the system. The layer above the hardware contains the core functionality of the system. It provides basic functionality for the SMG 2.0 system for further applications, including data aggregation, persistence and management of components and can be seen as a middleware solution. The third layer of SMG 2.0 contains the business logic. It is an EMS that includes components for automation, analysis and planing. The layer additionally provides several APIs for external applications, such as visualisations, market components, or external services. External applications have no access to the internal message brokering system and belong to the fourth layer.

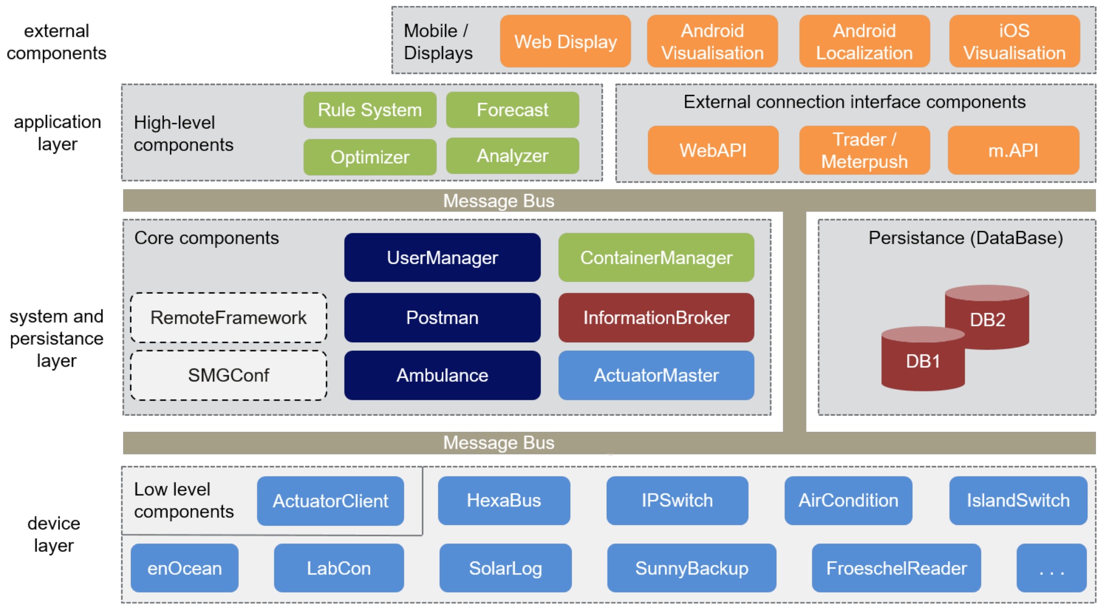


Figure A.2: Overview of SMG 2.0 software components

Hierarchy within the ContainerManager

Our system provides a solid base of functionality for the creation of various EMS applications. We start with the most important component for applications - the *ContainerManager*. It is used as a connection to the system's sensors and actuators. The *ContainerManager* provides the main API to access all recent information within the system and to control the actuators. It is designed according our generic hierarchic concept and allows to organize and structure the physical world entities, represented by the *ActuatorClients*, by their types, relationships and functionality with composites that we call container. A container in the system represents a part of the building that contains some connected devices. For instance a room is a container. It has a unique id and contains various sensors (or rather the information for subscription to corresponding *ActuatorClients*), actuators and other containers that represent further substructures, e.g. a work space of an employee or a complex device that contains multiple sensors/ actuators. Every container can be part of a higher hierarchical component, such as a room is part of a floor or a building. The hierarchical structure of the *ContainerManager* is shown in Figure A.3. The relationship between the containers is expressed by edges. It refers to the unique ids of the parent and the child. Based on those associations, the *ContainerManager* establishes the communication with the devices over the message brokering system.

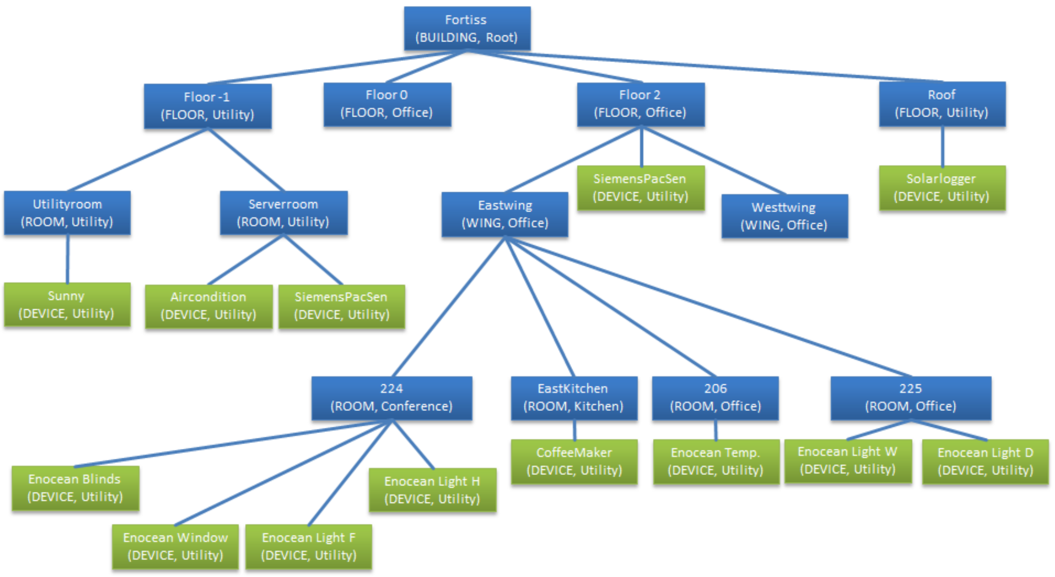

Fortiss (BUILDING, Root)
Floor -1 (FLOOR, Utility)
Floor 0 (FLOOR, Office)
Floor 2 (FLOOR, Office)
Roof (FLOOR, Utility)
Utilityroom (ROOM, Utility)
Serverroom (ROOM, Utility)
Eastwing (WING, Office)
SiemensPacSen (DEVICE, Utility)
Westtwing (WING, Office)
Solarlogger (DEVICE, Utility)
Sunny (DEVICE, Utility)
Aircondition (DEVICE, Utility)
SiemensPacSen (DEVICE, Utility)
224 (ROOM, Conference)
EastKitchen (ROOM, Kitchen)
206 (ROOM, Office)
225 (ROOM, Office)
CoffeeMaker (DEVICE, Utility)
Enocean Temp. (DEVICE, Utility)
Enocean Light W (DEVICE, Utility)
Enocean Light D (DEVICE, Utility)
Enocean Blinds (DEVICE, Utility)
Enocean Light H (DEVICE, Utility)
Enocean Window (DEVICE, Utility)
Enocean Light F (DEVICE, Utility)

Figure A.3: The generic concept of the *ContainerManager* component in SMG 2.0 .

The *ContainerManager* bundles the information received by the components. It listens to the events and aggregates them along the container structure in a bottom-up aggregation process. We have implemented different aggregation functions for every building parameter, such as lights, temperature and power usage. For lights we have multiple aggregations. We consider that lights are switched on, if at least one light switch is turned on, but we also provide a percentage value, that determines how much of the lightning is turned on. Beside that, we also have light-sensors that measure the current brightness. If multiple light-sensors are present, the aggregation is the average value. The averaging aggregation applies for temperature sensors as well. The aggregation function for the power consumption is a sum. Only the aggregated values are provided for applications that access the *ContainerManager* over its API and the message brokering system.

The downwards direction, from application to devices, is also handled by the *ContainerManager*. It provides the interface to control devices. For some devices we have a unified control mechanism, for instance for lights. Some of the devices are more complex. They have a unique specification. The control is enabled by forwarding the specification of their interface. This allows for applications to send specific control signals to the *ContainerManager*, which forwards them to the devices, and makes sure that their initialization is correct.

Another concept of the *ContainerManager* includes the usage of so called virtual containers. It is a concept to achieve higher flexibility to structure components. A virtual containers can freely combine different rooms and devices. The virtual container has the same capabilities as a regular container, however, it is not allowed to forward the values from the children upwards the hierarchy, to keep the aggregation functions intact. This increases the flexibility for applications, at the same time, the complexity remains controllable. Virtual containers can also be created at runtime and exist temporarily. This allows users to get control over office rooms, or some devices, only when necessary. At the same time, the values are not forwarded upwards. Hence, a virtual component receives all values such as the power consumption, however, they do not influence the overall consumption of the building. This keeps the complexity as simple as possible.

Furthermore, the *ContainerManager* allows a reconfiguration of components within the system. Any new ActuatorClient is initially connected to a container. The default container is the fortiss building. A user with access to the configuration of the *ContainerManager* can easily change the topology during runtime, by simply changing the edges between components. The *ContainerManager* recognizes those changes and makes sure that the whole behaviour of the system is adapted according the change, including the communication and the behaviour in terms of the aggregation functions.

Hierarchic architecture for the integration of multiple prosumers

The introduced concept and its implementation by the *ContainerManager* is not only applied to a single building automation system at fortiss, but is also successfully extended to distribution networks with remote locations and multiple SMG 2.0 systems, each with its own *ContainerManager* as well. Such a system is implemented in the Amrita Smart Grid Demonstrator[2] during the collaboration within the Indigo Stabilize-E Project (Gupta and Duchon, 2018) with the goal to improve the power supply reliability in local areas. We use multiple SMG 2.0 systems according our architecture concept to realize the flexible interconnection of EMS systems. The setting of the field test is shown in Figure A.4. Red boxes show the installed SMG 2.0 systems. They are responsible for the control of smart devices, which are illustrated by the blue boxes.

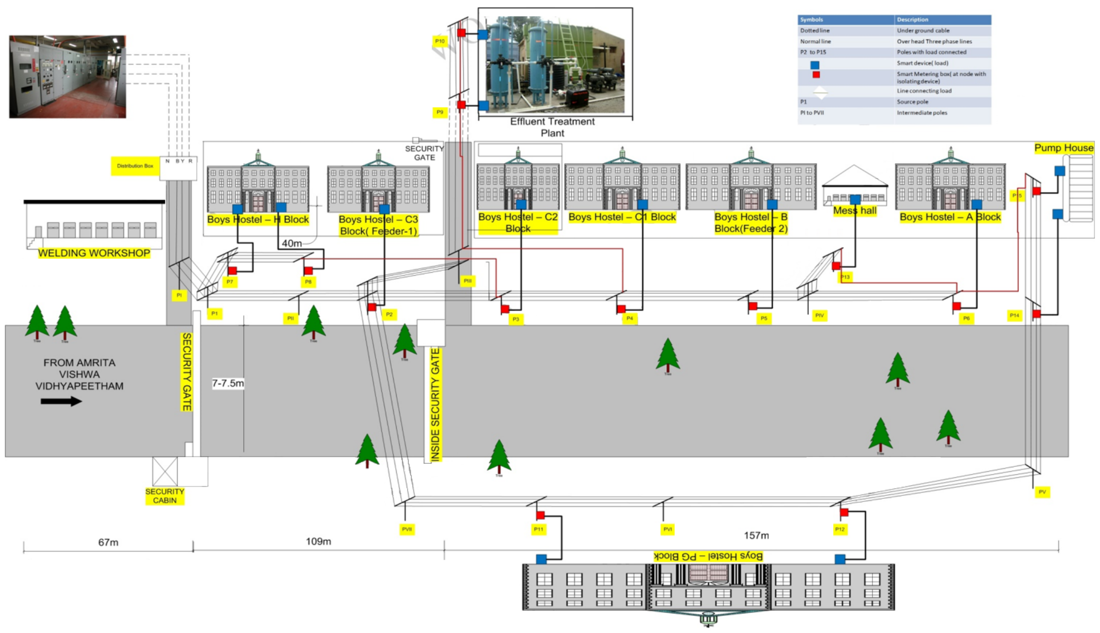


Figure A.4: Electrical scheme of the Amrita Smart Grid Demonstrator with several hierarchic SMG 2.0 prosumer systems based on (Gupta and Duchon, 2018).

[2]Amrita Vishwa Vidyapeetham (Amrita) - Center for Wireless Networks and Applications, Kerala, India, `https://www.amrita.edu/center/awna`, last accessed March 2020

The integration process is as following: An SMG 2.0 system registers another SMG 2.0 system as a child device. After registration, the parent system receives monitoring and control access to the aggregated functionality of the child SMG 2.0 . It gets only the aggregated values, without access to its internal structure and internal events. The hierarchical structure is shown in Figure A.5. The interconnection uses an *ActuatorClient* that establishes the connection via REST. Internally, the *ActuatorClient* registers the children as complex devices and receives a container ID from the *ContainerManager*. This allows to use the same *ContainerManager* specifications even for remote and more complex devices. As a result, hierarchic architecture is used twice in that field test, as an internal composition and an external one.

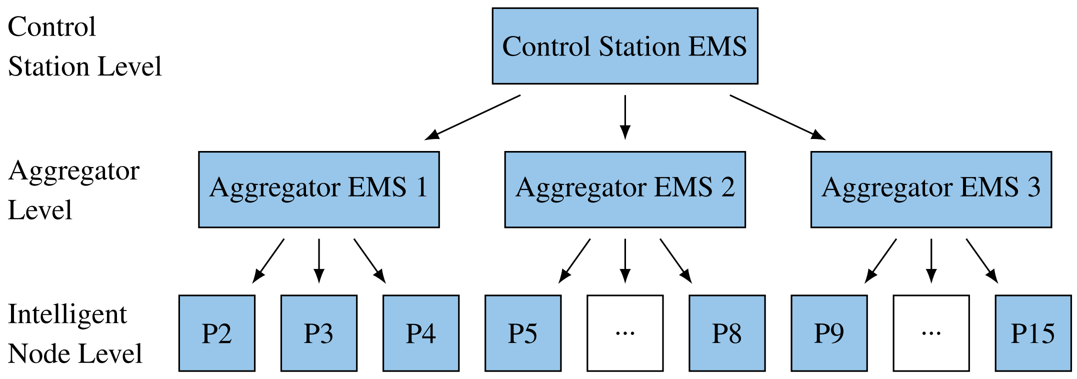


Figure A.5: Hierarchic composition of SMG 2.0 systems according (Gupta et al., 2016).

Further core components of the demonstrator

The other SMG 2.0 core components of the system have a support functionality. We shortly describe them here to establish an understanding how the whole system works. A more detailed description can be found in (Duchon et al., 2014).

The *InformationBroker* component represents the persistence of the data. It has access to a database system and stores every event that is sent over the message brokering system. We usually use a MySQL database on a larger server for this task. Occasionally, for some particular research activities we also use a specialized embedded database on small devices like RaspberryPis and Odroids, and also MongoDB for some specialized tasks. In addition to the storage functionality, the *InformationBroker* provides an interface to for accessing historical data. This is particularly important for higher level services. The *InformationBroker* has access to all system data, such a user profiles, configurations or historical event data.

The *ActuatorMaster* component represents the central registrar. Is us responsible to register all *ActuatorClients* from the lower layer. Whenever a new sensor or actuator component starts in the system, it registers at the actuator master with the specification of its capabilities. The master uses this specification to fulfil a number of tasks. First, it introduces the new client to the other core components, including *ContainerManager*, *InformationBroker*, *Ambulance* and *Postman*. From the introduction process, it receives additional information that it forwards to the client. This particularly includes a container id from the *ContainerManager*. After a successful registration, the clients can access the message brokering and send values or receive commands.

The *UserManager* controls the authentication and authorization process for components of users. It also manages the user profiles, which contain the desired settings, e.g. the room conditions such as preferable temperatures, temperature sensitivity and light conditions. Furthermore, it manages the user creation, deletion, or editing of users.

The *Postman* component is a notification system, that listens on all events and can generate different messages, such as email, twitter, or sms. This allows to implement any form of notifications that is desired by the system operator or system user.

The *Ambulance* component is responsible for the monitoring of the system's health and the management of the components. It checks which components are operational and which status they provide. If any component is malfunctioning, the *Ambulance* component can stop and restart the component. It can also generate events, so that the *Postman* components notifies the system operator.

Further, the SMG 2.0 core components rely on a set of common libraries. Libraries are utility functions to interact with components. The first one is the *RemoteFramework*. It handles the access to the messaging system. The library was designed to support different message bus systems. The current SMG 2.0 system uses the RabbitMQ[3] message broker with the Advanced Message Queuing Protocol (AMPQ) protocol. This allows a platform independent message exchange. The *RemoteFramework* supports also other messaging systems, like the OSGi internal message system, or the Java Message Service (JMS) and its implementation with Apache ActiveMQ[4]. The second library *SMGconf* is used for configuration. It contains deployment information like the addresses and ports of the server the system is running on, the database access and logging configuration.

The SMG 2.0 architecture is designed for a flexible system environment, where any component can be activated and stopped independently. We chose the OSGI platform for the implementation as our main environment. It runs on top of the JVM (Java Virtual Machine). The OSGI standard allows to implement components as bundles[5] that can be installed, started, stopped, updated and de-installed.

For further implementation details and a description of the other components we refer to the previously published material (Koß et al., 2012; Camek et al., 2013; Duchon et al., 2014; Gupta et al., 2015; Rottondi et al., 2015; Gupta et al., 2016; Gupta and Duchon, 2018).

[3]`https://www.rabbitmq.com/`, last accessed in August, 2019

[4]`http://activemq.apache.org/`, last accessed in August, 2019

[5]According the OSGi standard, a bundle is the smallest software unit of modularization. Technically, it is a .jar file with additional meta information.

B | Additional visualizations for the results in MEMAP

This appendix presents more details for the MEMAP case study, which were not shown in the previous chapter. Particularly, we show here the results of the optimization presented in Chapter 6 for each individual building. This represents the most efficient behaviour of a single building that has an own EMS system. It is the upper cost bar for each building and therefore the reference scenario to compare it with MEMAP's operation. The costs for every building were already presented in Table 6.1 for the fixed price and an cloudy summer day, and in Table 6.2 for the variable price and a sunny summer day. We structure the following of this chapter accordingly.

Fixed price, cloudy summer day

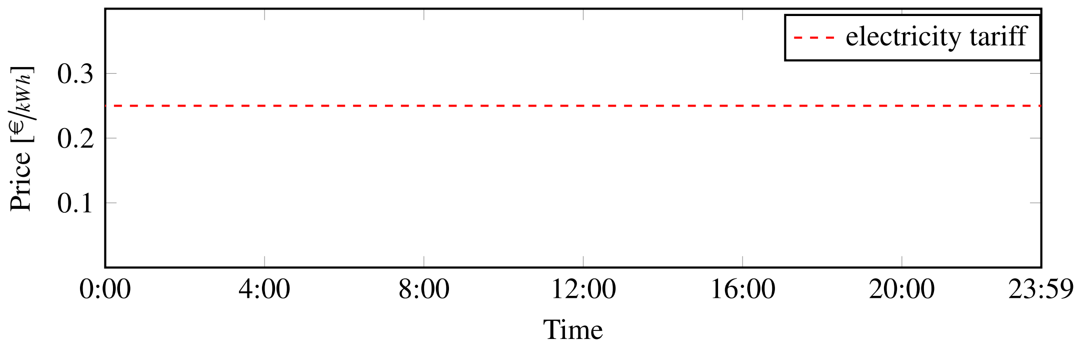


Figure B.1: The price signal for the scenario with the fixed price.

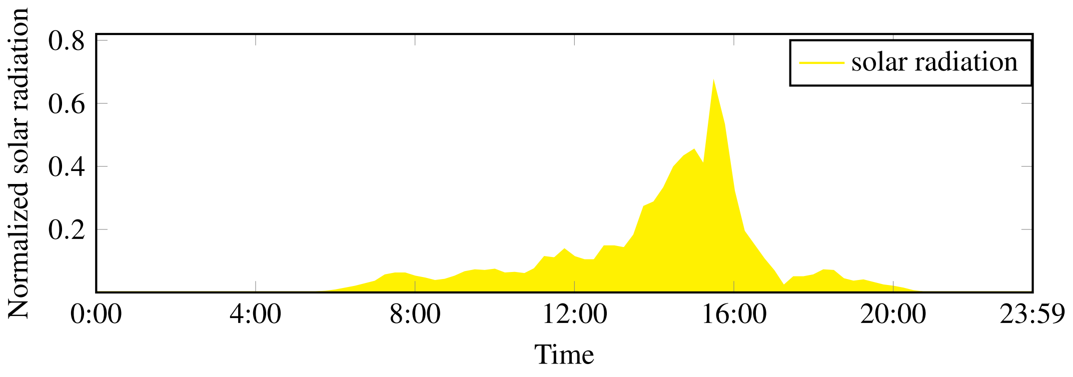


Figure B.2: The solar radiation power, cloudy summer day.

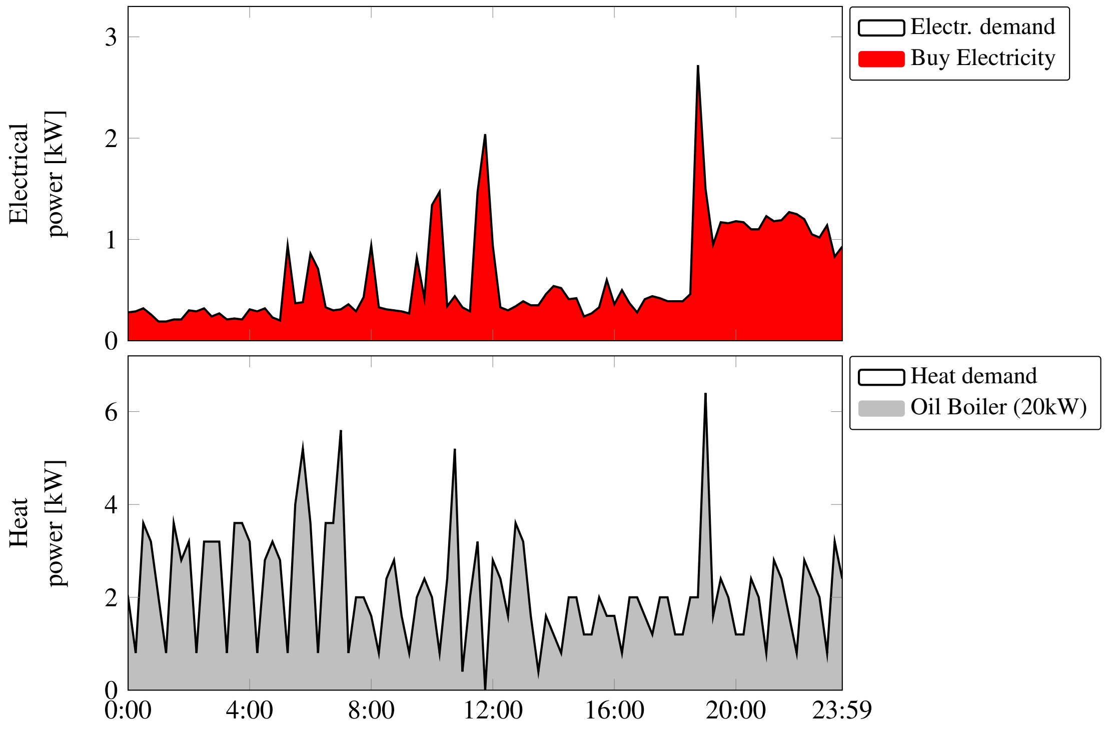


Figure B.3: Building #1, 6 hours MPC horizon, fixed price.

Building #1 has almost no flexibilities. Therefore it simply covers its thermal demand with its oil boiler and buys all electricity on the market. The oil boiler has an efficiency of 95% and costs of 6.85 [$^{ct}/_{kWh}$][1]. The building has a daily electricity demand of 14.4 kWh, a thermal demand for heating and hot-water of 51.4 kWh. The costs for that day are 7.42 €.

[1]Deutscher Energieholz- und Pellet-Verband e.V. (DEPV). Prices in Germany in August 2018. `https://depv.de/de/pelletpreis`, accessed in September 2018.

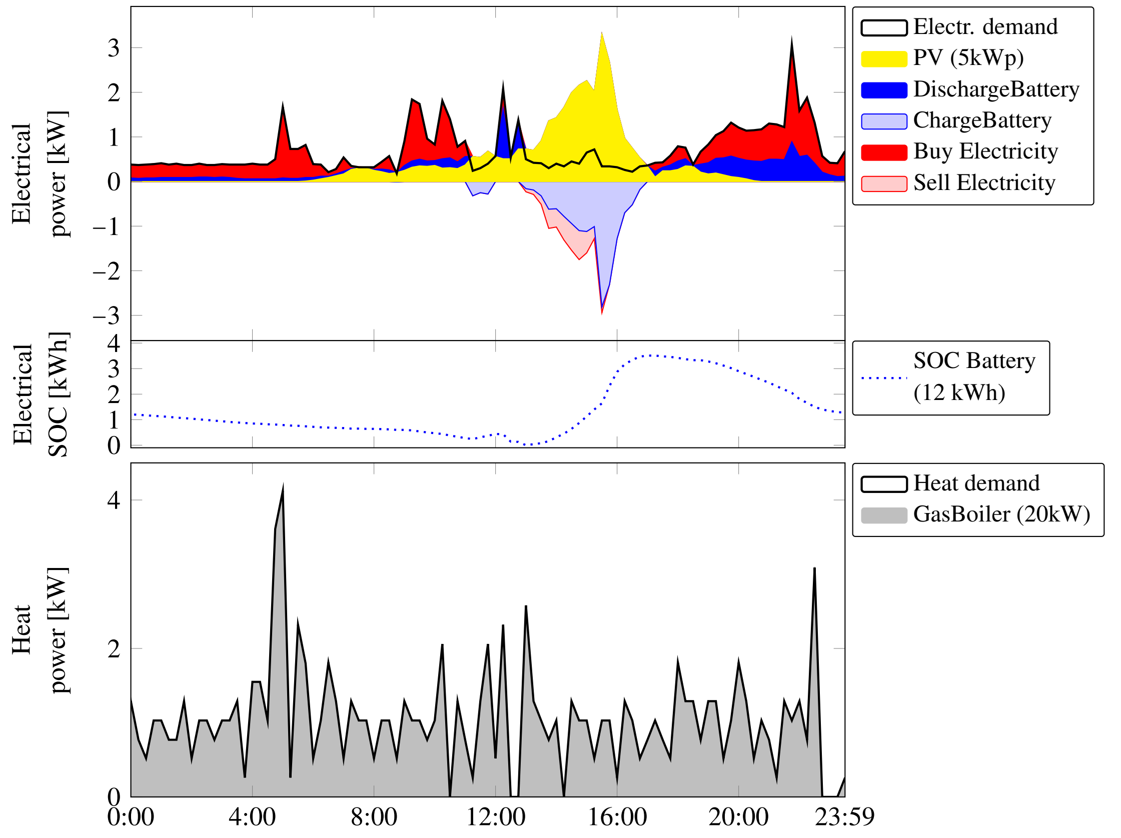


Figure B.4: Building #2, 6 hours MPC horizon, fixed price.

Building #2 has a similar thermal supply as the previous building but one additional photovoltaic system and a battery system. The optimization of its behaviour with a horizon length of 6 hours leads to a result, which stores enough power to cover a 6 hours period. The rest of the produced power is sold to the market. During the times without solar power the optimizations smooths the discharging such, that it continuously tries to discharge its battery in a 6 hours period. Therefore, it discharges the battery quite slowly, and buys the rest from the market. The gas boiler has an efficiency of 98% and costs of 5.91 $^{ct}/_{kWh}$. The demands for the day are 16.7 kWh electrical and 25.0 kWh thermal. The corresponding costs of building #2 are 3.59 €.

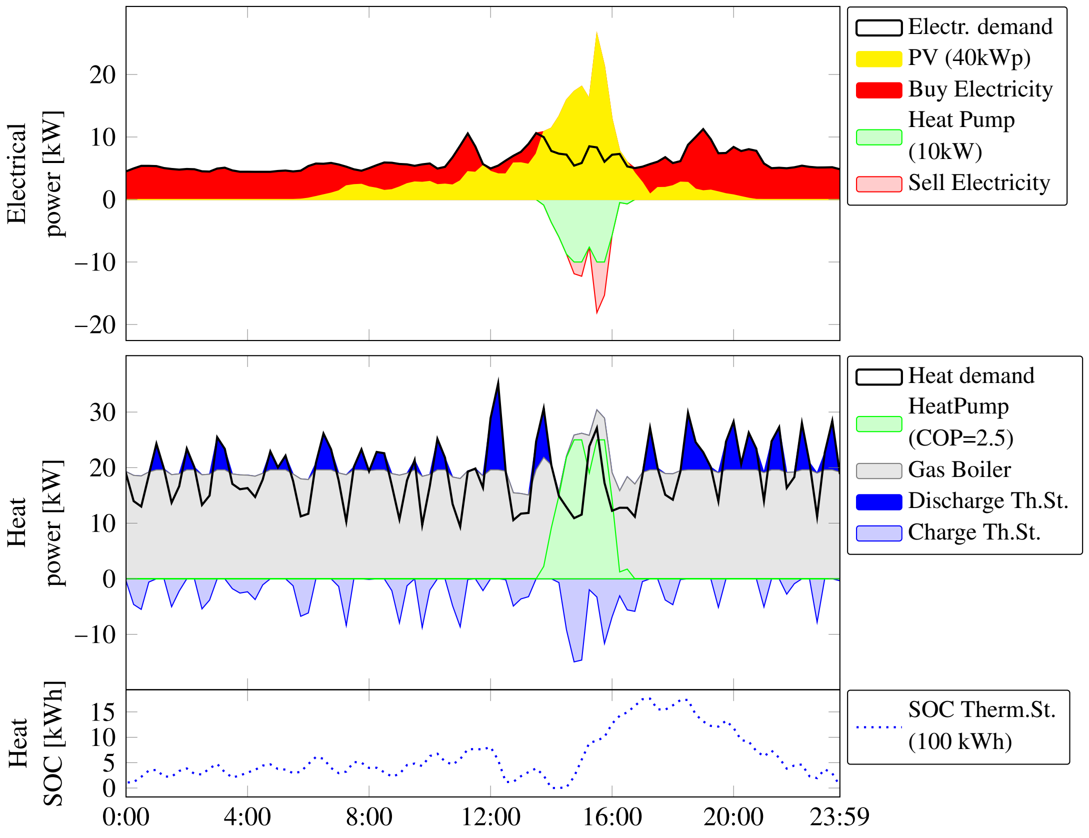


Figure B.5: Building #3, 6 hours MPC horizon, fixed price.

Building #3 has a 20 kW gas boiler with an efficiency of 98% and a 40 kWp photovoltaic system. In addition, it has a heat pump and thermal storage for its heat demand. The demands represent a multi family home. The optimization suggests to use the gas boiler at times, where the electrical demand is higher than the PV production. Because the gas boiler has a maximum heat production, which is often exceeded by the demand, the thermal storage is charged and discharged regularly. The reason for the preference of gas is that its production costs 6.03 $^{ct}/_{kWh}$ (i.e. 5.91 divided by 0.98), while the heat pump production with a COP of 2.5 and an electrical price of 25 $^{ct}/_{kWh}$ costs 10 $^{ct}/_{kWh}$. A better heat pump with a higher COP, or less expensive electricity costs could change this preference. During the sunny periods however, the heat pump becomes more economical than selling the the overproduction and starts with additional production. This charges the thermal storage as well. The overall costs of building #3 for that day are 48.44 €, to cover 145.8 kWh electrical and 457.1 kWh thermal demand.

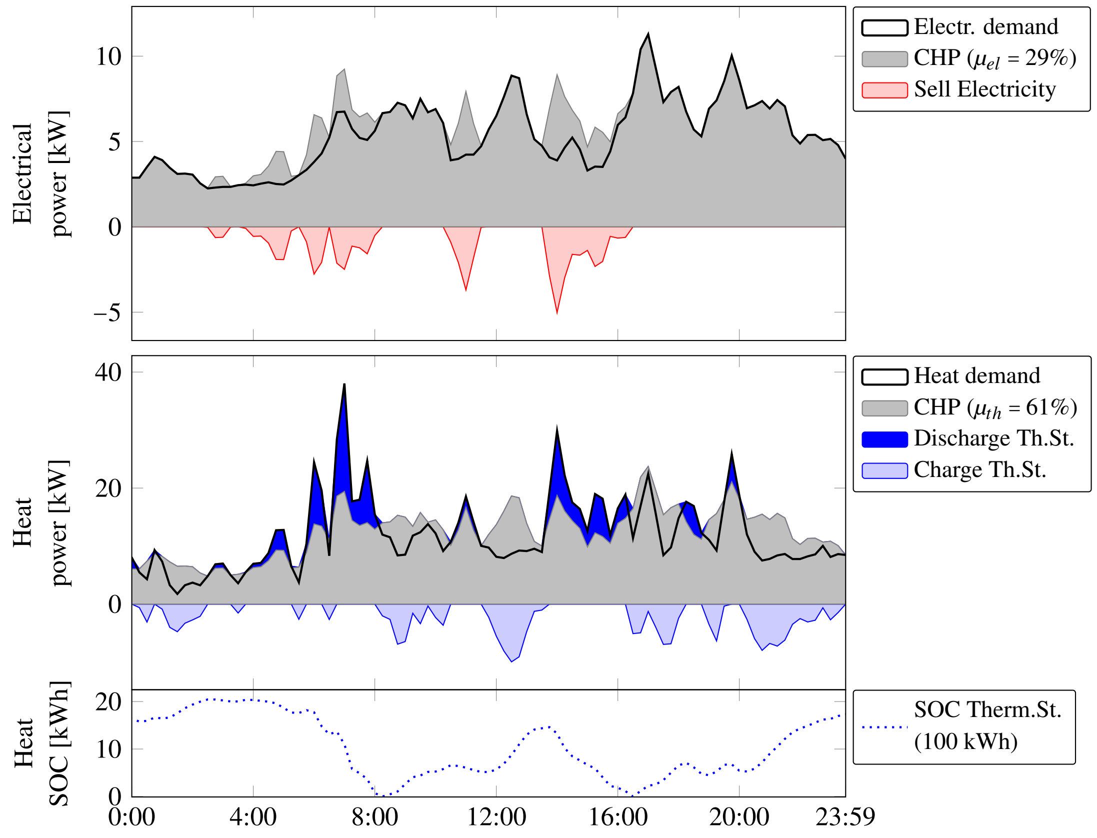


Figure B.6: Building #4, 6 hours MPC horizon, fixed price.

Building #4 has no solar production and no gas or oil boilers. Instead it has a CHP that can produce heat and electricity at the same time and a thermal storage to buffer heat. Its optimal operation shows that it avoids to buy electricity under any circumstances. It covers all of its electricity and stores the overproduced heat. In times with a very high thermal demand the CHP produces a lot and sells the overproduction of the electricity. Otherwise, there are not much to optimisation options left. The costs are 31.56 € for 127.5 kWh electricity and 286.5 kWh thermal demand.

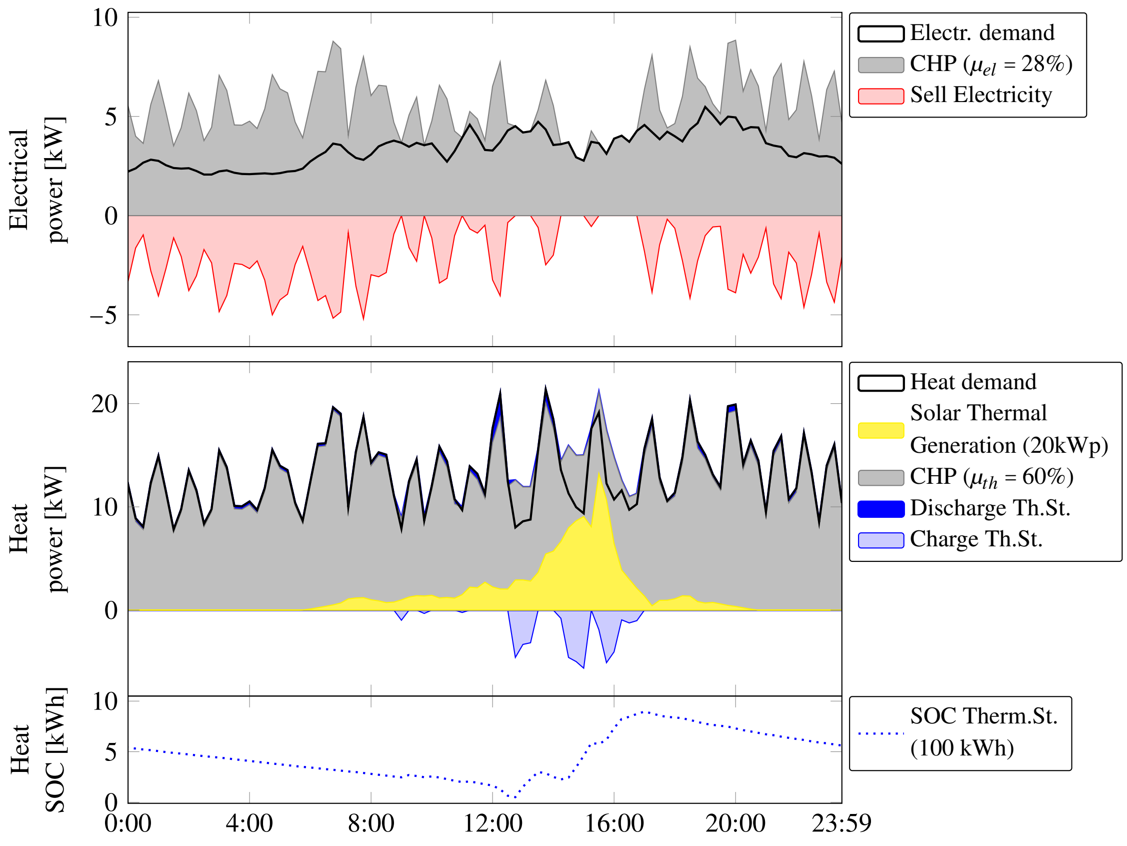


Figure B.7: Building #5, 6 hours MPC horizon, fixed price.

Building #5 is similar as building #4, but it has in addition one solar thermic generation unit with 20 kWp. Its daily demand is 80.5 kWh electric and 316.9 kWh thermal. This corresponds to more heat oriented demand profile and changes slightly the behaviour. Building #5 still controls its CHP in such way, so that it avoids any external buying. Hence, during sunny periods, with additional heat generation from solar thermal generation, it operates electricity driven. In the darker periods, it operates heat driven. The over production of electricity during the dark periods is sold to the market. The operational costs are slightly improved with solar thermal production and sum up to 25.82 € for that day.

Variable price, sunny summer day

To compare the fixed price scenario with a more future oriented situation, where prices are used as a signal to steer energy demands and supplies, we change the price profiles. We choose a simple price curve with two different price levels with a high price and a low price period, for a better understanding of the effect. Remember, that we sell the electricity with price that is 50% less compared to the current market prices, to consider taxes and additional dues. Already this simple price example shows that advanced EMS react to this external input. Additionally, to explore more of the storage systems potential, we assume a sunny day in this scenario.

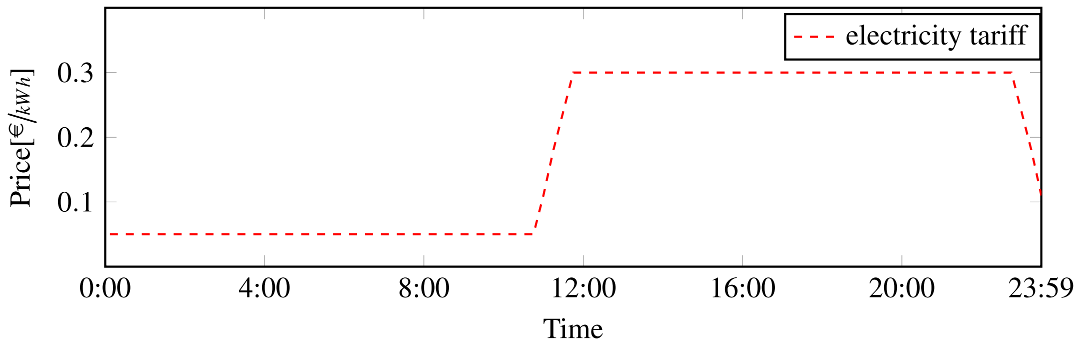


Figure B.8: The price signal for the scenario with the variable price.

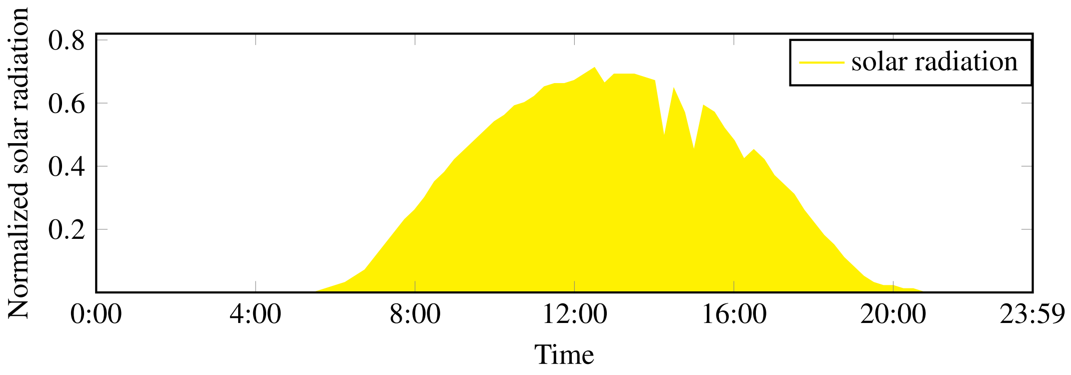


Figure B.9: The solar radiation power for a sunny summer day.

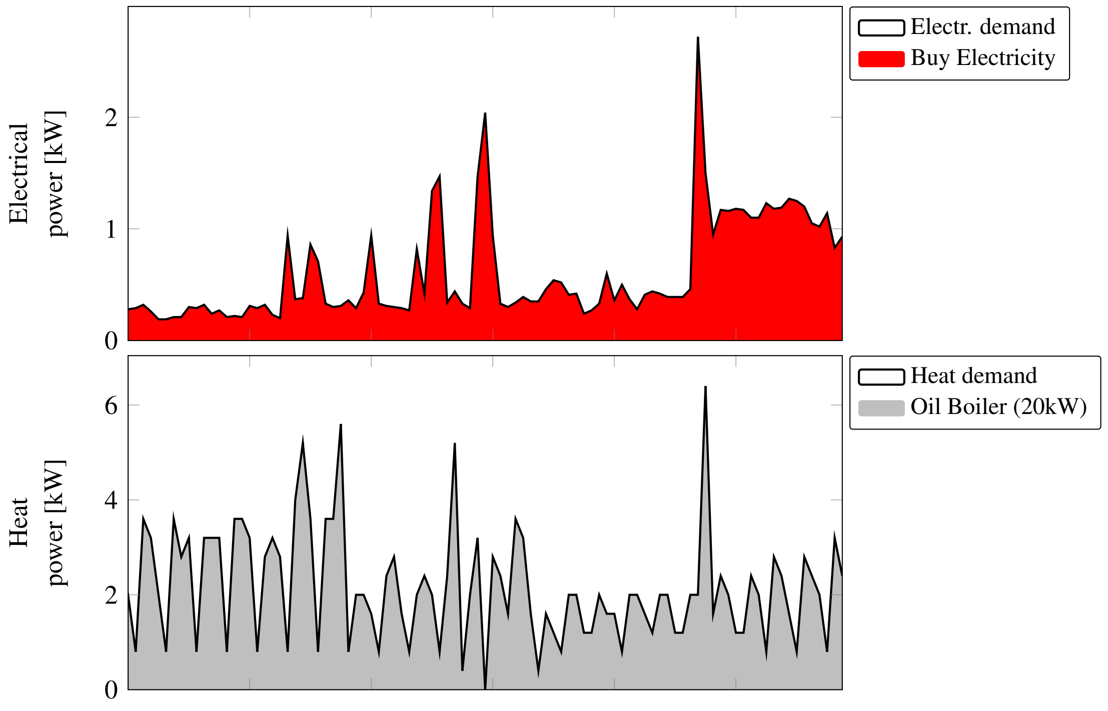


Figure B.10: Building #1, 6 hours MPC horizon, variable price.

Building #1 has the same behaviour in the flexible price scenario as before, because it has no flexibilities to adapt its behaviour and reduce the costs. The costs for that day are 9.89 €, due to the different electricity prices.

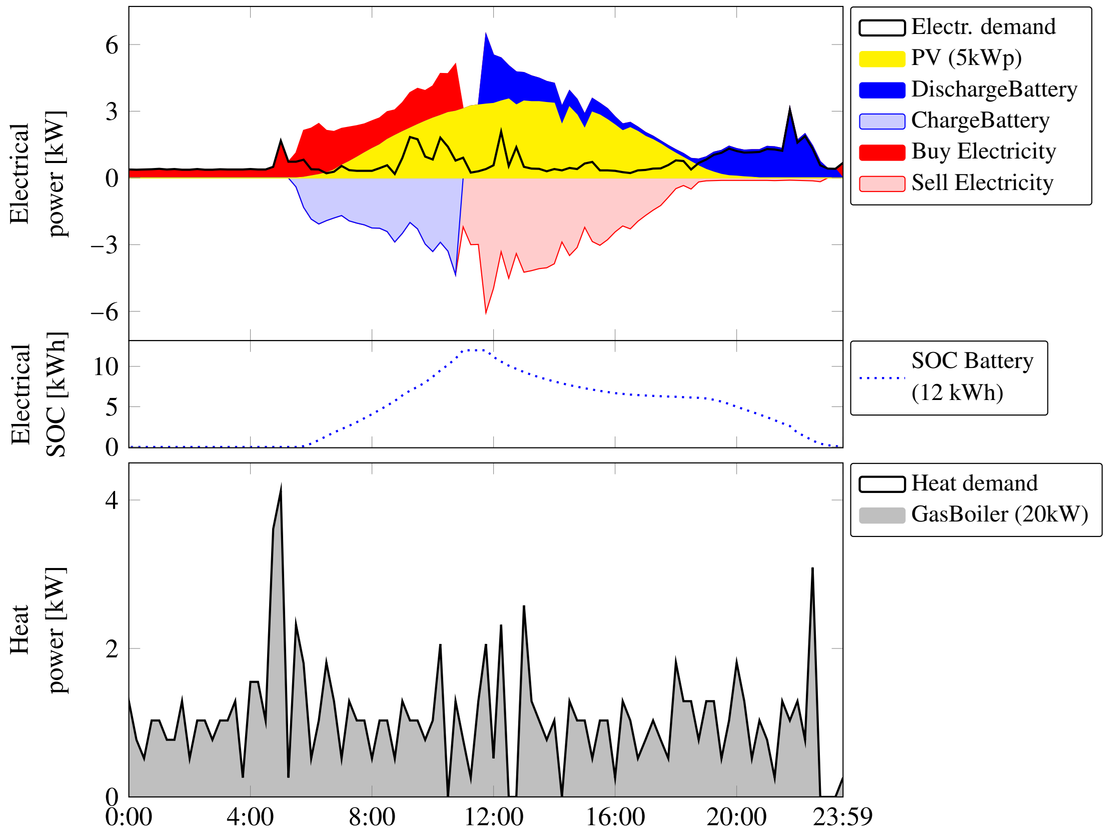


Figure B.11: Building #2, 6 hours MPC horizon, variable price.

Building #2 changes its behaviour completely for the second scenario. Firstly, its storage is completely used to reduce buying but particularly also for trading. The storage is fully charged at low price periods and discharged at the high price periods. The heat is covered by the gas boiler. The costs improves from 7.42 € in the first scenario to a negative value of -1.14 € in the second, due to the increased solar radiation and trading with the battery. Negative costs mean that the building has additional income.

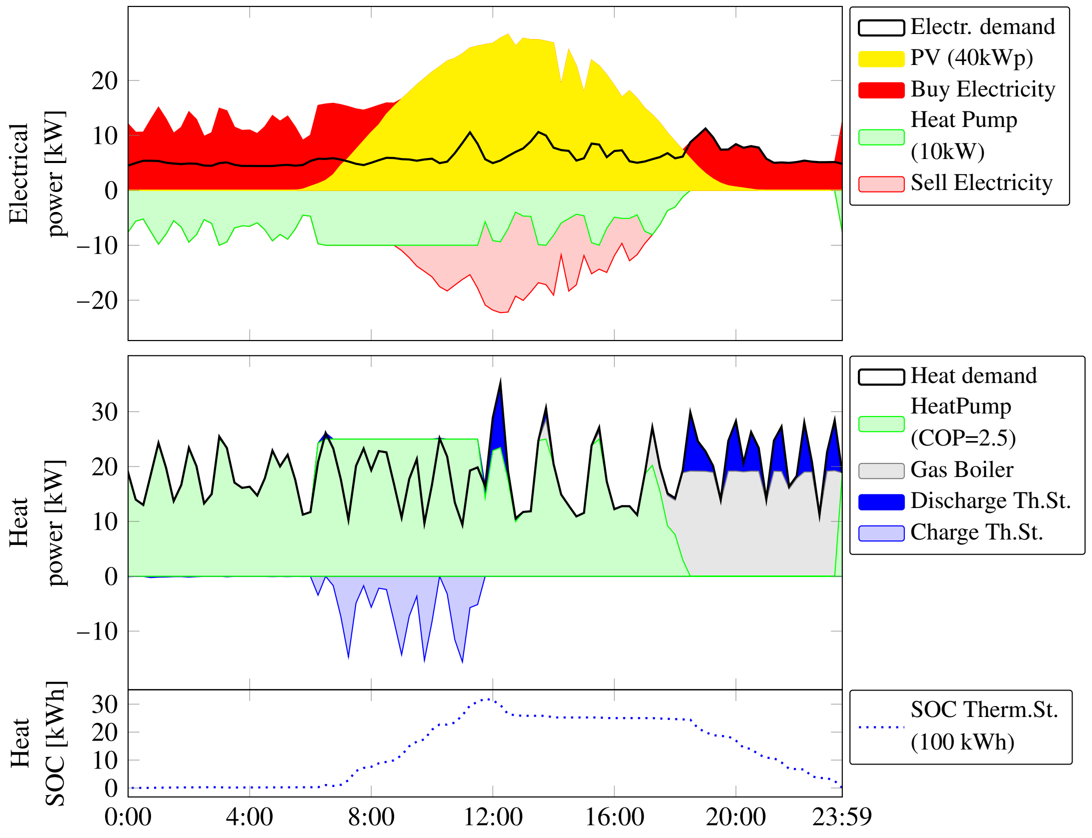


Figure B.12: Building #3, 6 hours MPC horizon, variable price.

Building #3 reacts on the variable price with its heat pump. During the low cost period, the heat pump has a cheaper production than the gas boiler and can therefore save costs. It tries to cover all the demand within the MPC horizon. In addition the heat pump is the preferred device during the sunny period, where it is also used to charge the thermal storage. When the price increases, the heat pump is only used to cover the heat demand, but not to produce heat for storage. Instead, the electricity from the PV is sold to the market. During the dark times and high prices, the electricity demand is bought from the market, while the heat demand is covered by the gas boiler and thermal storage. The costs for that day are 11.44 €, which is a significant improvement to the previous scenario, where they were more than four times higher.

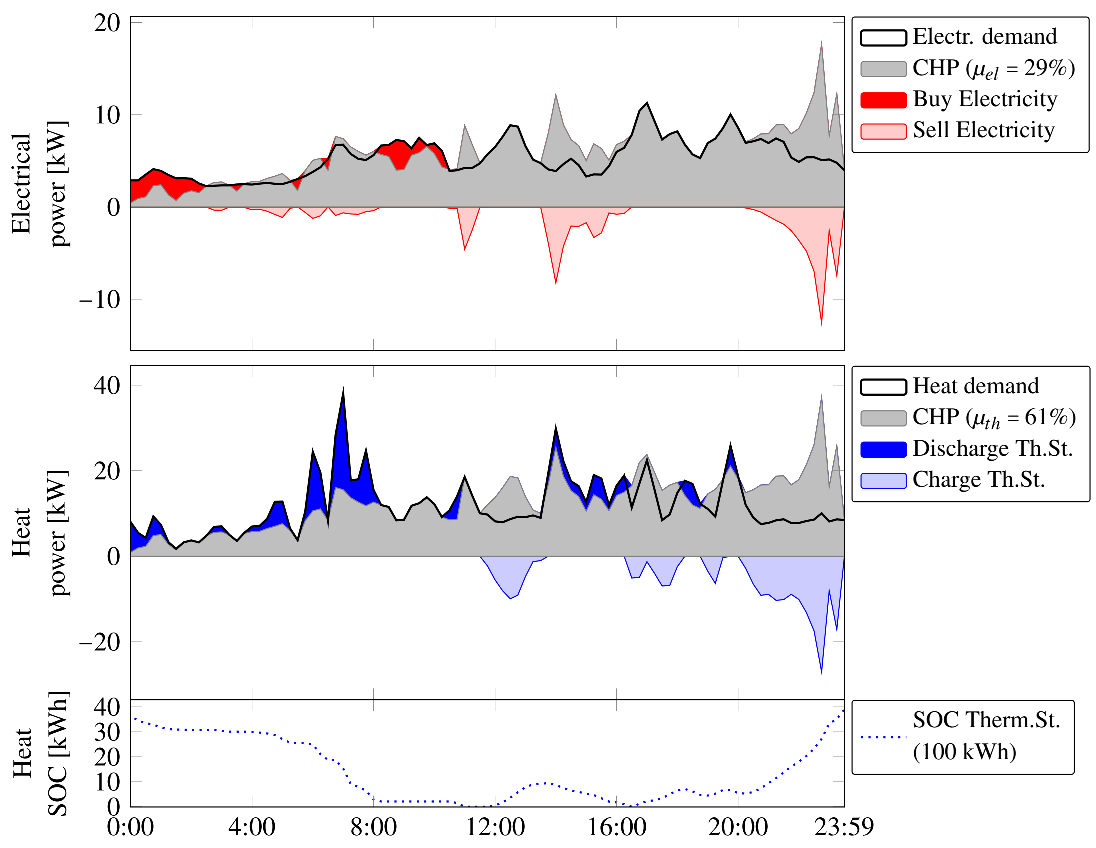


Figure B.13: Building #4, 6 hours MPC horizon, variable price.

Building #4 adapts its CHP usage to the variable price as well. During the low cost period it operates heat driven together with the heat storage to cover the thermal demand. The remaining electricity power is covered by the market. During the high cost period the CHP shifts to an electricity driven operation, that never buys electricity, sells the overproduced electricity, and supports with the heat demand with storage. When the MPC recognises an upcoming low cost period (next day has the same price profile), the CHP increases its production to store additional heat, which can be used during those low cost periods and reduce the CHP operation. The total costs for that day are 31.00 €, which is a only decent improvement to the previous scenario with 31.56 €.

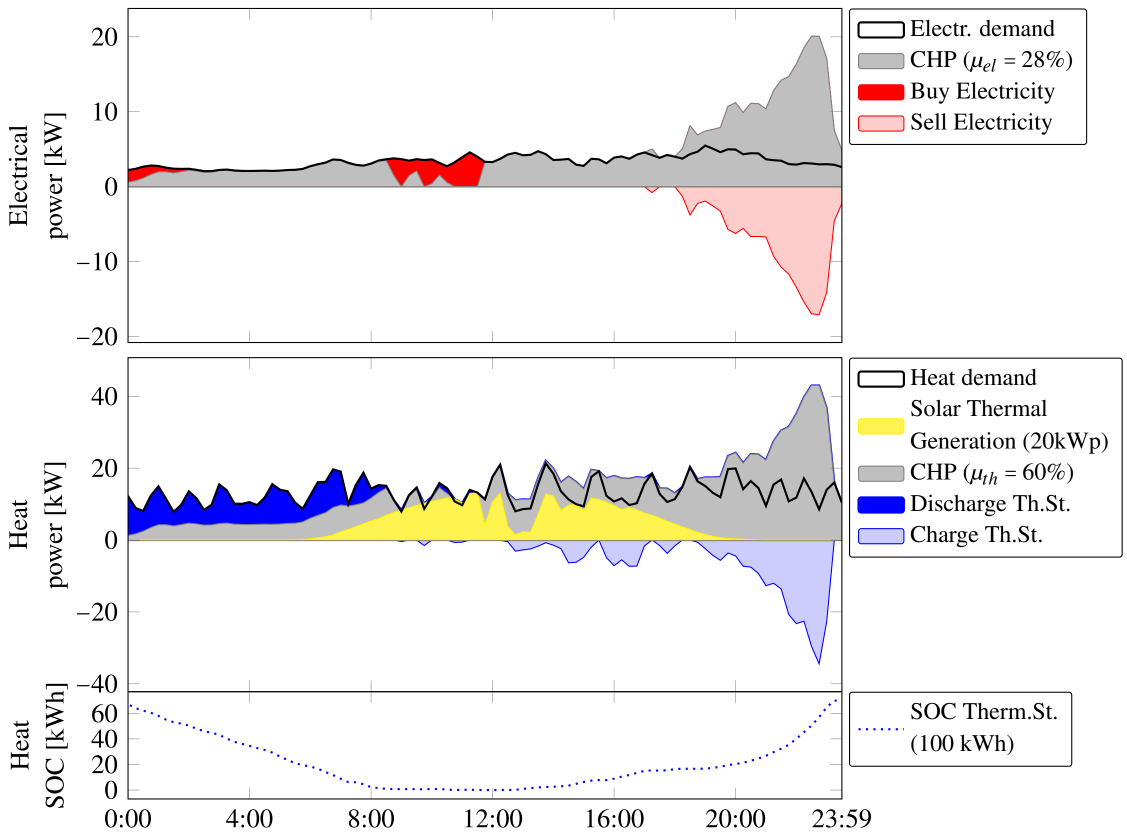


Figure B.14: Building #5, 6 hours MPC horizon, variable price.

Building #5 has similar schedules as building #4, meaning that it operates heat driven during the low cost period and uses its thermal storage as well. During the high cost period, it operates electricity driven and uses the MPC horizon to store enough heat for the next day. An interesting optimization result is that the freely available solar thermal heat production is not completely used, when the high cost period starts. The reason is that at 12:00 the optimization sees only high prices within the MPC horizon and wants to use the CHP to eliminate the need of buying electricity. Since the usage of the CHP produces heat as well, it reduces the solar thermal production. The costs of building #5 are 22.03 €, which is a decent reduction in comparison to the first scenario.

Short Summary

The second scenario with flexible prices shows that flexible buildings profit most. Photovoltaic systems and batteries are particularly useful. They help to avoid buying of electricity, use heat pumps and operate CHPs more heat driven. The reduction of costs is achieved by three major factors: increased solar radiation, batteries, and the availability of several components to choose the best option for each external condition. Particularly the last point is provided by MEMAP.

Bibliography

Acatech (2017). Sektorkopplung – Optionen für die nächste Phase der Energiewende. acatech – Deutsche Akademie der Technikwissenschaften und Akademie der Wissenschaften Leopoldina.

Agha, G. (1985). Actors: A model of concurrent computation in distributed systems. Technical report, MIT Computer Science and Artificial Intelligence Laboratory.

Akorede, M., Hizam, H., Aris, I., and Ab Kadir, M. (2010). A review of strategies for optimal placement of distributed generation in power distribution systems. *Research Journal of Applied Sciences*, 5(2):137–145.

Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., and Meedeniya, I. (2013). Software architecture optimization methods: A systematic literature review. *IEEE Transactions on Software Engineering*, 39(5):658–683.

Allerding, F. (2014). *Organic Smart Home Energiemanagement für Intelligente Gebäude*. Ph.D. Thesis, Karlsruhe Institute of Technology, KIT Scientific Publishing.

Andersson, G. (2004). Modelling and analysis of electric power systems. *EEH-Power Systems Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland*, pages 141–147.

Andersson, G. (2012). Power System Analysis. Course Compendium, ETH Zürich.

Andres, F. (2019). Machine Learning for Operational Optimization of District Heating Systems. Master's thesis, Technical University of Munich.

Andrews, G. R. (1991). Paradigms for process interaction in distributed programs. *ACM Computing Surveys (CSUR)*, 23(1):49–90.

Ascher, D. and Bytschkow, D. (2018). Integrating distribution system operator system landscapes. *Computer Science-Research and Development*, 33(1-2):169–175.

Asmus, P. (2010). Microgrids, virtual power plants and our distributed energy future. *The Electricity Journal*, 23(10):72–82.

AUTOSAR (2018). *AUTomotive Open System ARchitecture*. Standards collection: classic platform (for embedded systems); adaptive platform (for fail-operational systems); foundation (common parts); acceptance tests and application interfaces (description of application APIs). `www.autosar.org`.

Bajpai, A. (2018). Forecasting the Power Generation of Photovoltaic Cells using Machine Learning. Master Thesis, Technische Universität München.

Bajpai, A. and Duchon, M. (2019). Blockchain decision matrix for IoT systems. In *IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, University of British Columbia,Vancouver,Canada.

Baldick, R. (1995). The generalized unit commitment problem. *IEEE Transactions on Power Systems*, 10(1):465–475.

Baldick, R., Helman, U., Hobbs, B. F., and O'Neill, R. P. (2005). Design of efficient generation markets. *Proceedings of the IEEE*, 93(11):1998–2012.

Barnes, K., Johnson, B., and Nickelson, R. (2004). Review of supervisory control and data acquisition (SCADA) systems. Technical report, Idaho National Laboratory (INL).

Basso, G., Hilaire, V., Lauri, F., Paire, D., and Gaillard, A. (2013). A principled approach for smart microgrids simulation using MAS. In *German Conference on Multiagent System Technologies*, pages 193–207. Springer.

Becker, B. (2014). *Interaktives Gebaeude-Energiemanagement*. Ph.D. Thesis, Karlsruhe Institute of Technology, KIT Scientific Publishing.

Becker, B., Kern, F., Lösch, M., Mauser, I., and Schmeck, H. (2015). Building energy management in the FZI house of living labs. In *DACH Conference on Energy Informatics*, pages 95–112. Springer.

Benz, T., Dickert, J., Erbert, M., Erdmann, N., Johae, C., Katzenbach, B., Glaunsinger, W., Müller, H., Schegner, P., Schwarz, J., et al. (2015). Der zellulare Ansatz: Grundlage einer erfolgreichen, regionenübergreifenden Energiewende. *Studie der Energietechnischen Gesellschaft im VDE (ETG). Frankfurt am Main: VDE e. V.*

Blochwitz, T., Otter, M., Akesson, J., et al. (2012). Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference*, pages 173–184. Linköping University Electronic Press.

BMU (2016). Klimaschutzplan 2050. Klimaschutzpolitische Grundsätze und Ziele der Bundesregierung. Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit (BMU).

Braun, M. and Strauss, P. (2008). A review on aggregation approaches of controllable distributed energy units in electrical power systems. *International Journal of Distributed Energy Resources*, 4(4):297–319.

Bray, T. (2017). The JavaScript Object Notation (JSON) Data Interchange Format. Technical report, Internet Engineering Task Force (IETF).

Broy, M. and Stølen, K. (2001). *Specification and development of interactive systems: focus on streams, interfaces, and refinement*. Springer Science & Business Media.

Bundesnetzagentur (2018). Bundesnetzagentur ändert Zuschlagmechanismus bei Ausschreibung von Regelenergie. Bundesnetzagentur, Pressemitteilung.

Bytschkow, D. (2016). Towards composition principles and fractal architectures in the context of smart grids. *it-Information Technology*, 58(1):3–14.

Bytschkow, D. and Ascher, D. (2017). CIM, Domänenmodellierung und Herausforderungen für Softwaresysteme im integrierten Verteilnetzbetrieb. In *Zukünftige Stromnetze für erneuerbare Energien (OTTI)*.

Bytschkow, D., Capone, A., Mayer, J., Kramer, M., and Licklederer, T. (2019). An OPC UA based Energy Management Platform for Multi-Energy Prosumers in Districts. In *Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE PES.

Bytschkow, D. and Duchon, M. (2015). Ladestrategien für E-Fahrzeuge: Koordination vs. Optimierung. *ETG-Fachbericht-Von Smart Grids zu Smart Markets 2015*.

Bytschkow, D., Quilbeuf, J., Igna, G., and Ruess, H. (2014). Distributed MILS architectural approach for secure smart grids. In *International Workshop on Smart Grid Security*, pages 16–29. Springer, Cham.

Bytschkow, D., Zellner, M., and Duchon, M. (2015). Combining SCADA, CIM, GridLab-D and Akka for Smart Grid Co-Simulation. In *Innovative Smart Grid Technologies (ISGT)*. IEEE PES.

Camek, A., Hölzl, F., and Bytschkow, D. (2013). Providing security to a smart grid prosumer system based on a service oriented architecture in an office environment. In *Innovative Smart Grid Technologies (ISGT)*. IEEE PES.

Carpentier, J. (1962). Contribution a l'etude du dispatching economique. *Bulletin de la Societe Francaise des Electriciens*, 3(1):431–447.

Chrysoulas, C. and Fasli, M. (2017). Towards an adaptive SOA-based QoS & Demand-Response Provisioning Architecture for the Smart Grid. *JCOMSS*.

CIM (2015). Common Information Model Primer: Third Edition. Technical report, EPRI, Palo Alto, CA. 3002006001.

Contreras, J., Candiles, O., De La Fuente, J. I., and Gomez, T. (2001). Auction design in day-ahead electricity markets. *IEEE Transactions on power Systems*, 16(1):88–96.

Daneels, A. and Salter, W. (1999). What is SCADA? In *International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 339–343.

Dänekas, C., Neureiter, C., Rohjans, S., Uslar, M., and Engel, D. (2014). Towards a model-driven-architecture process for smart grid projects. In *Digital enterprise design & management*, pages 47–58. Springer.

Deka, D., Backhaus, S., and Chertkov, M. (2017). Structure learning in power distribution networks. *IEEE Transactions on Control of Network Systems*, 5(3):1061–1074.

Dijkstra, E. W. (1982). *On the Role of Scientific Thought*, pages 60–66. Springer New York.

Duchon, M., Gupta, P. K., Koss, D., Bytschkow, D., Schätz, B., and Wilzbach, S. (2014). Advancement of a sensor aided smart grid node architecture. In *IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 349–356.

EEBus (2018). EEBus SPINE (Smart Premises Interoperable Neutral-message Exchange) standard. Technical report, EEBUS Initiative e.V.

ENTSO-E (2018). Bidding zone configuration. Technical report, European Network of Transmission System Operators for Electricity.

ESO (2012). *The M/490 Mandate Smart Grids Reference Architecture*. European Standards Organizations.

Esslinger, P. and Witzmann, R. (2012). Entwicklung und Verifikation eines stochastischen Verbraucherlastmodells für Haushalte. In *12. Symposium Energieinnovation in Graz*.

Fall, K. R. and Stevens, W. R. (2011). *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley.

Fan, J. and Borlase, S. (2009). The evolution of distribution. *Power and Energy Magazine, IEEE*.

Fang, X., Misra, S., Xue, G., and Yang, D. (2012). Smart grid - the new and improved power grid: A survey. *Communications Surveys & Tutorials, IEEE*, 14(4):944–980.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine.

Frey, S. (2013). Generic architectures for open, multi-objective autonomic systems: application to smart micro-grids. PhD Thesis, Télécom ParisTech.

Frey, S., Diaconescu, A., Menga, D., and Demeure, I. M. (2013). A holonic control architecture for a heterogeneous multi-objective smart micro-grid. *IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 21–30.

Gamma, E., Richard, H., Johnson, R. E., and Vissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Garlan, D., Allen, R., and Ockerbloom, J. (1995). Architectural mismatch or why it's hard to build systems out of existing parts. In *Proceedings of the 17th international conference on Software engineering*, pages 179–185. ACM.

Garlan, D. and Shaw, M. (1993). An introduction to software architecture. *Advances in software engineering and knowledge engineering*, pages 1–39.

Geidl, M. and Andersson, G. (2007). Optimal power flow of multiple energy carriers. *IEEE Transactions on power systems*, 22(1):145–155.

Geidl, M., Koeppel, G., Favre-Perrod, P., Klockl, B., Andersson, G., and Frohlich, K. (2007). Energy hubs for the future. *IEEE Power and Energy Magazine*, 5(1):24–30.

Ghavidel, S., Li, L., Aghaei, J., Yu, T., and Zhu, J. (2016). A review on the virtual power plant: Components and operation systems. In *IEEE International Conference on Power System Technology (POWERCON)*. IEEE.

Glück, B. (1984). *Heizwassernetze für Wohn- und Industriegebiete*. VEB Verlag für Bauwesen.

Godfrey, T., Mullen, S., Griffith, D. W., Golmie, N., Dugan, R. C., and Rodine, C. (2010). Modeling smart grid applications with co-simulation. In *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 291–296. IEEE.

Grijalva, S., Costley, M., and Ainsworth, N. (2011). Prosumer-based control architecture for the future electricity grid. In *IEEE International Conference on Control Applications (CCA)*, pages 43–48. IEEE.

Grijalva, S. and Tariq, M. U. (2011). Prosumer-based smart grid architecture enables a flat, sustainable electricity industry. In *Innvative Smart Grid Technologies (ISGT)*. IEEE PES.

Gupta, P. and Duchon, M. (2018). Developing self-similar hybrid control architecture based on sgam-based methodology for distributed microgrids. *Designs*, 2(4):41.

Gupta, P. K., Gibtner, A. K., Duchon, M., Koss, D., and Schätz, B. (2015). Using knowledge discovery for autonomous decision making in smart grid nodes. In *IEEE International Conference on Industrial Technology (ICIT)*, pages 3134–3139. IEEE.

Gupta, P. K., Martins, R., Chakarbarti, S., Remanidevi, A. D., Mäki, K., Krishna, G., Schätz, B., Ramesh, M. V., and Singh, S. N. (2016). Improving reliability and quality of supply (QoS) in smart distribution network. In *National Power Systems Conference (NPSC)*, pages 1–6.

Hackenberg, G., Irlbeck, M., Koutsoumpas, V., and Bytschkow, D. (2012). Applying formal software engineering techniques to smart grids. In *International Workshop on Software Engineering for the Smart Grid (SE4SG)*, pages 50–56. IEEE.

Haller, P. (2012). On the integration of the actor model in mainstream technologies: the scala perspective. In *Proceedings of the 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions*. ACM.

Haller, P. and Odersky, M. (2009). Scala actors: Unifying thread-based and event-based programming. *Theoretical Computer Science*, 410:202–220.

Hashmi, M., Hanninen, S., and Maki, K. (2011). Survey of smart grid concepts, architectures, and technological demonstrations worldwide. In *In Innovative Smart Grid Technologies Latin America (ISGT LA)*. IEEE PES.

Heidemann, L., Bytschkow, D., Capone, A., Licklederer, T., and Kramer, M. (2019). Sector coupling with optimization: A comparison between single buildings and combined quarters. In *The 8th DACH+ Conference on Energy Informatics*.

Hewitt, C. and Baker, H. (1977). Laws for communicating parallel processes. *AI Working Paper*.

Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *Advance Papers of the Conference*, volume 3, page 235. Stanford Research Institute.

Hopkinson et. al., K. (2006). Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. *IEEE Transactions on Power Systems*.

Howell, S., Rezgui, Y., Hippolyte, J.-L., Jayan, B., and Li, H. (2017). Towards the next generation of smart grids: Semantic and holonic multi-agent management of distributed energy resources. *Renewable and Sustainable Energy Reviews*, 77:193–214.

Huber, M. (2017). *Flexibility in Power Systems - Requirements, Modeling, and Evaluation*. PhD thesis, Technische Universität München.

IEEE (2011). Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads. Ieee, IEEE.

Irlbeck, M., Bytschkow, D., Hackenberg, G., and Koutsoumpas, V. (2013). Towards a bottom-up development of reference architectures for smart energy systems. In *2nd International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG)*. IEEE.

Irlbeck, M. and Koutsoumpas, V. (2015). Die E-Energy Referenzarchitektur. Project Report, Technische Universität München.

Jambagi, A., Kramer, M., and Cheng, V. (2015). Residential electricity demand modelling: Activity based modelling for a model with high time and spatial resolution. In *3rd International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE.

Jokić, A. (2007). *Price-based optimal control of electrical power systems*. PhD thesis, Technische Universiteit Eindhoven.

Junker, M. (2016). *Specification and Analysis of Availability for Software-Intensive Systems*. PhD thesis, Dissertation, München, Technische Universität München, 2016.

Karmani, R. K., Shali, A., and Agha, G. (2009). Actor frameworks for the jvm platform: a comparative analysis. In *Proceedings of the 7th International Conference on Principles and Practice of Programming in Java*, pages 11–20. ACM.

Keddis, N. (2015). *Capability-based Planning and Scheduling of Workflows for Adaptable Manufacturing Systems*. PhD thesis, Technical University of Munich.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 1:41–50.

Kirschen, D. S. and Strabac, G. (2004). *Fundamentals of Power System Economics*. John Wiley and Sons Ltd.

Koestler, A. et al. (1967). *The ghost in the machine*. Hutchinson London.

Koß, D., Bytschkow, D., Gupta, P. K., Schätz, B., Sellmayr, F., and Bauereiß, S. (2012). Establishing a smart grid node architecture and demonstrator in an office environment using the soa approach. In *International Workshop on Software Engineering for the Smart Grid (SE4SG)*. IEEE.

Kramer, M., Jambagi, A., and Cheng, V. (2016). A model predictive control approach for demand side management of residential power to heat technologies. In *International Energy Conference (ENERGYCON)*. IEEE.

Kramer, M., Jambagi, A., and Cheng, V. (2017). Distributed model predictive control for building energy systems in distribution grids. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE PES.

Kundur, P., Balu, N. J., and Lauby, M. G. (1994). *Power system stability and control*, volume 7. McGraw-hill New York.

Lange, J., Iwanitz, F., and Burke, T. (2010). *OPC: Von Data Access bis Unified Architecture*. 4., völlig neu bearbeitete und erweiterte Auflage, VDE Verlag GMBH, Berlin.

Lasseter, R. H. (2011). Smart distribution: Coupled microgrids. *Proceedings of the IEEE*, 99(6):1074–1082.

Lasseter, R. H. and Piagi, P. (2004). Microgrid: A conceptual solution. In *IEEE Power Electronics Specialists Conference*, volume 6, pages 4285–4291. Citeseer.

Lee, E. A., Neuendorffer, S., and Wirthlin, M. J. (2003). Actor-oriented design of embedded hardware and software systems. *Journal of circuits, systems, and computers*, 12(03):231–260.

Lightbend (2018). *Documentation of Akka*. `https://akka.io/docs/`.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin et. al, H. (2011). Power system and communication network co-simulation for smart grid applications. In *ISGT*. IEEE PES.

Liskov, B. (1988). Data abstraction and hierarchy. *SIGPLAN notices*, 23(5):17–34.

Liu, X. and Mancarella, P. (2016). Modelling, assessment and sankey diagrams of integrated electricity-heat-gas networks in multi-vector district energy systems. *Applied Energy*, 167:336–352.

Lumani, E. (2018). Decentralized Energy Trading in Microgrids through Blockchain and Smart Contracts. Master's thesis, Technical University of Munich.

Lund, H., Werner, S., Wiltshire, R., Svendsen, S., Thorsen, J. E., Hvelplund, F., and Mathiesen, B. V. (2014). 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems. *Energy*, 68:1–11.

Lund, P., Cherian, S., and Ackermann, T. (2005). A cell controller for autonomous operation of a 60 kv distribution area. *International Journal of Distributed Energy Resources*, 2(2):83–100.

Mack, A. (2014). Development of an agent-based simulation framework for smart grids. Master Thesis, Technical University of Munich.

Mancarella, P. (2014). MES (multi-energy systems): An overview of concepts and evaluation models. *Energy*, 65:1–17.

Marmsoler, D. (2018). Hierarchical specification and verification of architectural design patterns. In Russo, A. and Schürr, A., editors, *Fundamental Approaches to Software Engineering*, pages 149–168, Cham. Springer International Publishing.

Mauser, I. (2017). *Multi-modal Building Energy Management*. PhD thesis, Karlsruher Institut für Technologie (KIT). 37.06.01; LK 01.

McMorran, A. W., Ault, G. W., Elders, I. M., Foote, C. E., Burt, G. M., and McDonald, J. R. (2004). Translating cim xml power system data to a proprietary format for system simulation. *IEEE Transactions on Power Systems*, 19(1):229–235.

Milano, F. (2008). Continuous newton's method for power flow analysis. *IEEE Transactions on Power Systems*, 24(1):50–57.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.

Molitor, C., Groß, S., Zeitz, J., and Monti, A. (2014). Mescos—a multienergy system cosimulator for city district energy systems. *IEEE Transactions on Industrial Informatics*, 10(4):2247–2256.

Moslehi, K. and Kumar, R. (2010). A reliability perspective of the smart grid. *IEEE Transactions on Smart Grid*, 1(1):57–64.

Murali, D., Rajaram, M., and Reka, N. (2010). Comparison of facts devices for power system stability enhancement. *International Journal of Computer Applications*, 8(4):30–35.

Müsgens, F., Ockenfels, A., and Peek, M. (2014). Economics and design of balancing power markets in germany. *International Journal of Electrical Power & Energy Systems*, 55:392–401.

Nalini, B. K., Eldakadosi, M., You, Z., Zade, M., Tzscheutschler, P., and Wagner, U. (2019). Towards prosumer flexibility markets: A photovoltaic and battery storage model. In *Innovative Smart Grid Technologies (ISGT)*.

Negeri, E., Baken, N., and Popov, M. (2013). Holonic architecture of the smart grid. *Smart Grid and Renewable Energy*, 4(02):202.

Neubeck, P. R. (2012). *A probabilistic theory of interactive systems*. PhD thesis, Technische Universität München.

Neureiter, C. (2017). A domain-specific, model driven engineering approach for systems engineering in the smart grid. MBSE4U.

NIST (2010). NIST framework and roadmap for smart grid interoperability standards, release 1.0. National Institute of Standards and Technology.

Nosratabadi, S. M., Hooshmand, R.-A., and Gholipour, E. (2017). A comprehensive review on microgrid and virtual power plant concepts employed for distributed energy resources scheduling in power systems. *Renewable and Sustainable Energy Reviews*, 67:341–363.

Nussbaumer, T., Thalmann, S., Jenni, A., and Ködel, J. (2017). Planungshandbuch fernwärme. Technical report, EnergieSchweiz, Bundesamt für Energie BFE.

OPCFoundation (2017). OPC Unified Architecture Specification. Part 1: Overview and Concepts.

OpenADR (2013). Open Automated Demand Response (OpenADR 2.0b Profile Specification). Technical report, OpenADR Alliance.

Orehounig, K., Evins, R., and Dorer, V. (2015). Integration of decentralized energy systems in neighbourhoods using the energy hub approach. *Applied Energy*, 154:277–289.

Overbye, T. J., Cheng, X., and Sun, Y. (2004). A comparison of the ac and dc power flow models for lmp calculations. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. IEEE.

Papazoglou, M. (2003). Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE)*, pages 3–12.

Papazoglou, M. P. and Van Den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*.

Perry, D. E. and Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model predictive control: Theory and design*. Nob Hill Pub.

Reuter, A. and Breker, S. (2018). C/sells: Netze und märkte verbünden! Positionspapier anlässlich des Ministerdialogs am 7. August 2018.

Richter et. al, U. (2006). Towards a generic observer/controller architecture for organic computing. *GI Jahrestagung (1)*, 93:112–119.

Rittel, H. W. and Webber, M. M. (1974). Wicked problems. *Man-made Futures*, 26(1):272–280.

Rohjans, S., Dänekas, C., and Uslar, M. (2012). Requirements for smart grid ict-architectures. In *3rd Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE PES.

Rohjans, S., Lehnhoff, S., Schütte, S., Scherfke, S., and Hussain, S. (2013). Mosaik: A modular platform for the evaluation of agent-based smart grid control. In *4th Innovative Smart Grid Technologies Europe (ISGT EUROPE)*. IEEE PES.

Rottondi, C., Duchon, M., Koss, D., Palamarciuc, A., Pití, A., Verticale, G., and Schätz, B. (2015). An energy management service for the smart office. *Energies*, 8(10):11667–11684.

Santjer, F., Teichmann, K., and Steinert, W. (2002). Basics and conception of a Virtual-Powerplant in Germany. *DEWI Magazine*, 20:37–42.

Schaber, K., Steinke, F., and Hamacher, T. (2012). Transmission grid extensions for the integration of variable renewable energies in europe: Who benefits where? *Energy Policy*, 43:123–135.

Schermeyer, H. (2018). *Netzengpassmanagement in regenerativ geprägten Energiesystemen*. PhD thesis, Karlsruher Instituts für Technologie (KIT).

Schmeck, H. (2005). Organic computing-a new vision for distributed embedded systems. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*. IEEE.

Schmeck et. al., H. (2010). Adaptivity and self-organization in organic computing systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Schütte, S. (2013). *Simulation Model Composition for the Large-Scale Analysis of Smart Grid Control Mechanisms*. PhD thesis, Citeseer.

Schütte, S., Scherfke, S., and Tröschel, M. (2011). 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)*, pages 55–60. IEEE.

Shaw, M. and Clements, P. (1997). A field guide to boxology: Preliminary classification of architectural styles for software systems. In *Computer Software and Applications Conference, 1997. COMPSAC'97. Proceedings., The Twenty-First Annual International*, pages 6–13. IEEE.

Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22(3):183–236.

Sigloch, H. (2017). *Technische Fluidmechanik*. Springer. 10. aktuallisierte Auflage.

Simoglou, C. K., Biskas, P. N., and Bakirtzis, A. G. (2010). Optimal self-scheduling of a thermal producer in short-term electricity markets by MILP. *IEEE Transactions on Power Systems*, 25(4):1965–1977.

Sinteg (2018). Förderprogramm SINTEG: Schaufenster intelligente Energie - Digitale Agenda für die Energiewende. Press release. BMWI.

Steinbrink, C., Blank-Babazadeh, M., El-Ama, A., Holly, S., Lüers, B., Nebel-Wenner, M., Ramírez Acosta, R. P., Raub, T., Schwarz, J. S., Stark, S., et al. (2019). Cpes testing with mosaik: Co-simulation planning, execution and analysis. *Applied Sciences*, 9(5):923.

Stouffer, K., Falco, J., and Kent, K. (2006). Guide to supervisory control and data acquisition (scada) and industrial control systems security. *Recommendations of the National Institute of Standards and Technology*.

SWM (2015). Fernwärme und Rücklauftemperatur in modernen Niedertemperaturnetzen. Fernwärmeversorgung in München. *ArtikelNr. 116822.*

Söder, L. (2011). Analysis of electricity markets. Course compendium, KTH Stockholm.

Söder, L. and Amelin, M. (2011). Efficient operation and planning of power systems. Course compendium, KTH Stockholm.

Taylor, R. N., Medvidovic, N., and Dashofy, E. M. (2009). *Software architecture: foundations, theory, and practice*. Wiley Publishing.

Sauter Cumulus GmbH (2018). Sauter Building and Energy Management System Architecture, *Internal Project Deliverable of MEMAP (used with permission).*

Wikimedia Commons (2006). Map of european transmission system operators organizations.

Thut, A. (2018). Development and evaluation of a DLT-based marketplace for sector coupling in quarters. Master Thesis, Technical University of Munich.

Tinney, W. F. and Hart, C. E. (1967). Power flow solution by newton's method. *IEEE Transactions on Power Apparatus and systems*, pages 1449–1460.

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. (2011). Observation and control of organic systems. In *Organic Computing—A Paradigm Shift for Complex Systems*, pages 325–338. Springer.

Uslar, M., Specht, M., Rohjans, S., Trefke, J., and González, J. M. (2012). *The Common Information Model CIM: IEC 61968/61970 and 62325 - A practical introduction to the CIM*. Springer Science & Business Media.

VHPready (2017). Virtual Power Plant (VPP): Communication path between control center (CC) and distributed energy resource (DER). Technical report, Industry Alliance VHPready e.V.

Vogel-Heuser, B., Kegel, G., Bender, K., and Wucherer, K. (2009). Global information architecture for industrial automation. *Automatisierungstechnische Praxis (atp)*, 51(1):108–115.

Vogelsang, A. (2015). *Model-based requirements engineering for multifunctional systems*. PhD thesis, Technische Universität München.

Wesche, J., Dütschke, E., and Friedrichsen, N. (2017). Entstehung innovativer Wärmenetze – Eine Analyse von sechs Fallbeispielen auf Basis der Multi-Level-Perspektive. Technical report, Faunhofer ISI.

Willems, B. (2005). Physical and financial virtual power plants. *Tilburg University - Department of Economics, Available at SSRN 808944.*

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Wolff, D. and Jagnow, K. (2011). Überlegungen zu Einsatzgrenzen und zur Gestaltung einer zukünftigen Fern- und Nahwärmeversorgung. Technical report, Ostfalia Hochschule.

Zade, M., Incedag, Y., El-Baz, W., Tzscheutschler, P., and Wagner, U. (2018). Prosumer integration in flexibility markets: A bid development and pricing model. In *2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–9. IEEE.

Zellner, M. (2014). A simulation plattform for smart grids. Bachelor Thesis, Technical University of Munich.

Zimmermann, H. (1980). OSI reference model – The ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*.